CYBERTANK ENGINEER'S HANDBOOK

Neural Cybertank Design and Simulation

OMEGA

CONTENTS

＊＊＊取扱い説明編＊＊＊

＊＊＊エンジニアズ・ハンドブック編＊＊＊

PART2

PART3

PART4

PART5

PART6

取扱い説明編

★ はじめに

この度は“ＯＭＥＧＡ”をお買い上げいただきまして、誠に有難うございます。
“ＯＭＥＧＡ”は、近未来の主力戦車サイバータンクのＡＩ（人工知能）をユーザー自身がプログラミングし、これをコンピュータの戦車と戦わせて、プログラミング・テクニックを競うという画期的かつ究極のシミュレーション・ゲームです。
マニュアルに記載されたノウハウを理解し、これを駆使してＡＩプログラムを作成することは大変な努力の伴う作業ですが、それだけに勝利したときの喜びは格別のものと言えるでしょう。
皆さんも、“Slow but steady”の精神でこのマニュアルを読みこなし、“ＯＭＥＧＡ”が持つ無限の潜在力を味わって頂きたいと思います。

★ 製品の内容確認

“ＯＭＥＧＡ”のパッケージのなかには、以下のものが含まれます。
プレイを始める前に内容をご確認下さい。

★	ＯＭＥＧＡ“システム・ディスク”	１枚
★	ＯＭＥＧＡ“ＩＤディスク”	１枚
★	CYBERTANK ENGINEER'S HANDBOOK	１部
★	ＯＭＥＧＡアンケートはがき	１枚

万一不足品のございました場合は、お買い上げのお店、もしくは巻末に記載しました“トンキンハウス「ユーザー・サポート係」”までお問い合わせ下さい。

★ “OMEGA” の起動、及び初期設定

★ 起動

端末に電源を入れたら、OMEGAの “システム・ディスク” をいずれかのドライブにセットし、リセット・ボタンを押します。

※ 動作に異常のある場合は、p.20 をご参照下さい。

★ 入力モード の設定（キーボード／マウス）

上のタイトル画面を⏎キーでクリアすると “キーボード入力” に左クリックでクリアすると “マウス入力” に、それぞれ自動的に設定されます。画面は次頁の “セキュリティ・クリアランス” に転送されます。

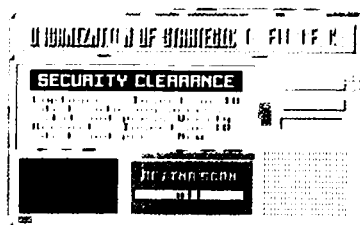
★ ディスプレイ・モードの設定（アナログ／液晶）

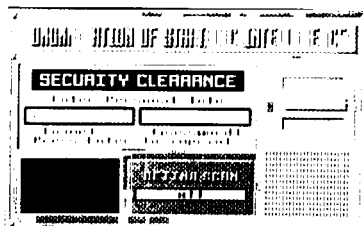
ディスクが端末の仕様を判別して自動的に設定します。但し、次のキーを押しながら起動すると強制的に設定することもできます。この設定はゲームに入ってから変更することも可能です。(p.34)

CTRL キー ……… アナログ
SHIFT キー ……… 液晶

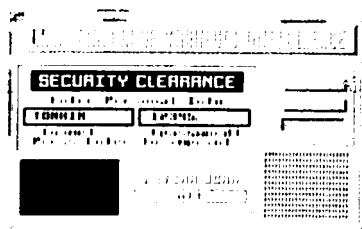
※ PC-98UR，NCをご使用の方はアナログに設定して下さい。

★　ＩＤ登録

次は“ＩＤディスク”に“登録名”と“パスワード”を記録してＩＤ登録を行ないます。この手続きを済ませた後は、パスワードを入力しないと、ＩＤディスク内のファイルにアクセスすることはできません。ＩＤ登録は, “セキュリティ・クリアランス” (Security Clearance)と呼ばれる場所で行なわれます。

Security Clearance

1) Security Clearance　に入ったら、ここでＩＤディスクを空いているドライブにセットします。１ドライブ・マシーンをご使用の方は、システム・ディスクをＩＤディスクに差し替えて下さい。
   ＩＤディスクをセットしたら、[New] を選んで下の画面に入ります。
   (キーボードの場合) [↑],[↓]キーで☞を [New] に合わせて[↵]キー。
   (マウスの場合) [New] をクリック。

2) ここで、キーボードから登録名とパスワードを入力します。

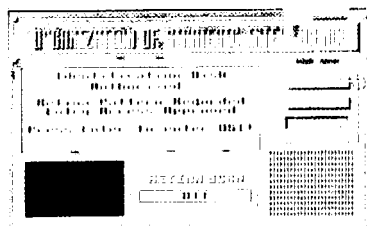
- 登録名、及びパスワードには英数文字（大文字、小文字どちらでも可）を使用し、最大8文字までとします。
- [カナ] キーがロックされていると入力できません。
- 打ち間違えたときは [BS] キーか [DEL] キーで修正します。

操作は次の通りです。
まず登録名を入力し、入力を終えたら [Enter] を選択してカーソルをパスワードの欄に移動させます。
（キーボードの場合）☞を [Enter] に合わせて⏎キー。
（マウスの場合）⏎キーを押すか、または [Enter] をクリック。
なお、カーソルを登録名の欄に戻すときは、[TAB] キーを押します。
次にパスワードを入力し、再び [Enter] を選択します。

※ パスワードは一旦登録してしまうと、それを照会する手段はありませんので、忘れないようどこかに書きとめておくか、もしくは覚え易いものにしておきましょう。

3) 画面に“＊＊＊ Recognized”という表示が出ると、次に網膜スキャナーが登録者の容姿を取り込んで、ＩＤディスクにインプリントします。

Enter を選択したら、身体を動かさないようにして網膜スキャナーを見つめ、インプリンティングの作業が終わるのを待ちます。

4) 下の画面に切り替わったら、ＩＤ登録完了です。

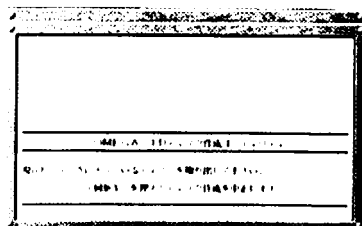
Enter を選択して、OMEGA SYSTEM にアクセス。画面は直ちに、External Control Module に転送されます。(p.34 参照)

※ ＩＤ登録とは、自分が作成したデータ・ファイルを専門に納めるディレクトリをＩＤディスク上に開設するという作業を意味し、“登録名”がそのディレクトリ名となります。

※ １枚のＩＤディスクには複数のユーザーがＩＤ登録をすることができます。
但し、登録名が重複すると、新しいディレクトリが上書きされ、古いディレクトリ内のファイルは全て失われてしまいますので、注意して下さい。

★　ＩＤディスクの作成

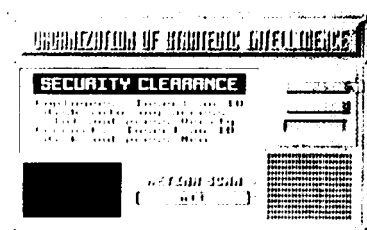
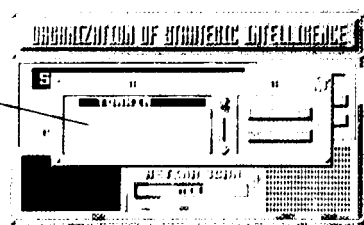
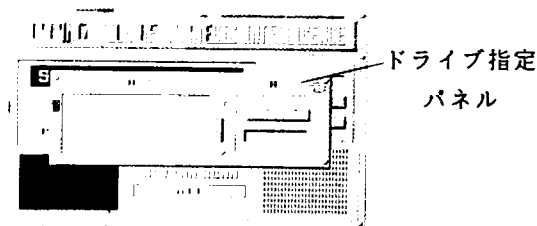
製品版とは別に新しくＩＤディスクを作成する場合は“ＩＤディスク作成ユーティリティ”を起動してこれを行ないます。

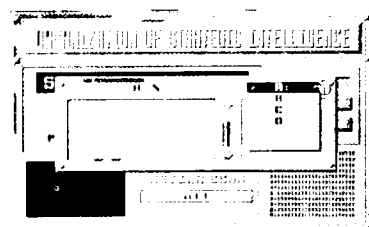
1）まず、２ＨＤの空ディスクを１枚用意します。

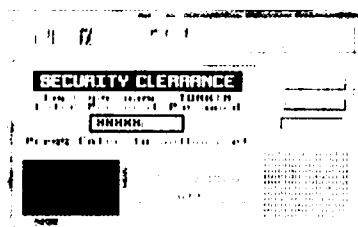
2）システム起動直後、トンキン・ハウスのロゴ・マークが表示されたところで ESC キーを押し、“ＩＤディスク作成ユーティリティ”を起動します。以後の操作は画面の指示に従って下さい。

※ＩＤディスクの作成は、画面の指示にもある通り、システムを立ち上げたドライブのみを使って、ディスク差し替えによって行ないます。従って２ドライブ・マシーンをご使用の方もＩＤディスクの作成に限っては、いずれか一方のドライブだけをご使用下さい。

★　IDディスクへのアクセス

既にIDディスクの作成とID登録を済ませている場合は、以下の手順に従ってIDディスクにアクセスします。

1) システム・ディスクで“OMEGA”を立ち上げます。
2) セキュリティ・クリアランスに入ったら、ここでIDディスクをセットします。
　(2ドライブ・マシーンをご使用の場合)
　空いているドライブにIDディスクをセットします。
　(1ドライブ・マシーンをご使用の場合)
　システム・ディスクを抜いて、IDディスクに差し替えます。

3) [Verify] を選択したら、次に [Enter] を選択して、ディレクトリ・ウィンド(下図参照)を開きます。
　(キーボードの場合) [↑],[↓]キーで☞を項目に合わせて[↵]キー。
　(マウスの場合) 項目を直接クリック。

※ “OMEGA” はこのとき、IDディスクのセットされているドライブを自動的にサーチして、そこのディレクトリを表示するよう設計されていますが、下のように、システム・ディスクの入っているドライブにアクセスしてしまった場合は、“ドライブ指定パネル” を使ってカレント・ドライブをIDディスクのセットされているドライブに変更します。

(キーボードの場合) ☜をドライブ指定パネルに合わせて⏎キーを押します(下図の状態)。次に ↑,↓キーで☜をIDディスクの入っているドライブに合せてこれ反転させ、⏎キーを押します。

(マウスの場合) ドライブ指定パネルをクリックしたまま☜を動かし(下図の状態)、IDディスクの入っているドライブが反転したところでクリック・ボタンを離します。

4) 自分の登録名が反転しているのを確かめてから、[Open] を選びます。
(キーボードの場合) [↑],[↓]キーで☞を [Open] に合わせて↵キー。
(マウスの場合) [Open] をクリック。
但し、複数のディレクトリが存在して、自分の登録名が反転していない場合は、以下の操作によってこれを反転させます。
(キーボードの場合) [←],[→]キーで☞をディレクトリ・ウィンドに移し、[↑],[↓]キーで自分の登録名を反転させる。
(マウスの場合) 自分の登録名をクリックする。

5) キーボードから自分のパスワードを入力し、[Enter] を選択します。

※ 入力したパスワードは秘密保持のため、画面には全て “XXXX” のかたちで表示されます。

間違ったパスワードを入力してエラーの警報が鳴った場合は、
(キーボードの場合) ☞を [Enter] に置いて↵キーを２回押して、
(マウスの場合) [Enter] をダブル・クリックして、
3) からもう１度やり直します。

6) “＊＊＊ Recognized” の表示が出たら、[Enter] を選択。

7) 網膜スキャニングが終わったら [Enter] を選択。 画面は直ちに External Control Module に転送されます。(p.34 参照)

★ 動作に異常のあるときは

ディスクが正常に動作しないときは、以下のことをお確かめ下さい。

● 動作しない、或いは途中で止まってしまう場合

☆ご使用の機種は、ＮＥＣ／ＰＣ－９８０１シリーズＶＭ以降のものでしょうか? “ＯＭＥＧＡ”は以下の機種には対応しておりません。

ＰＣ９８０１(ノーマーク)，Ｅ，Ｆ，Ｍ，Ｕ，ＸＡ，ＬＴ，
及び、ＮＥＣ以外の互換機種

☆本体メモリは６４０Ｋバイトになっているでしょうか?

・ディップスイッチＳＷ３の６番は“オフ”(メモリ６４０Ｋ使用)になっていますか?

・ＰＣ９８０１ＶＦ，ＶＭ，ＵＶでは拡張メモリが必要です。

● “このディスクからは起動できません”という表示が出た場合

☆ご使用のドライブはＮＥＣ製でしょうか?
他メーカーのドライブでは動作しない場合があります。

☆コピーしたディスクをお使いではないでしょうか?
コピーしたものはご使用になれません。

● 表示画面の色がおかしい

"OMEGA" のディスプレイ・モードは、アナログ(16色)と液晶8階調の2つに対応しており、起動時にディスクがマシーンの仕様を判別して自動的にこれを設定します。
これが適切に設定されなかった場合は、p.12 もしくは p.34 の操作でこれを変更して下さい。特にラップトップをご使用の方はアナログ/液晶と反転表示の組み合わせで、最も見やすいものに設定して下さい。

● キーボード入力が効かない

[カナ] キーがロックされていると、登録名、パスワード、コマンドの入力はできません。

"テンキー" はカーソル移動専用になっていますので、数字を入力する場合は、フルキー上列の数字キーをご使用下さい。

以上の項目を全てご確認のうえ、それでもなおディスクが正常に動作しない場合はご面倒ですが、最終頁の "お問い合わせ用紙" に必要事項を漏れなくご記入のうえ、これを切りとって当該ディスクと一緒に巻末に記載しました "トンキンハウス「ユーザー・サポート係」" までご送付いただくか、またはお電話にてお問い合わせ下さい。

※ 但し、ゲームの内容に関するお問い合わせには、お答えできませんので御了承下さい。

★　ディスクの有償交換について

万一お客様の過失によりシステム・ディスクを破損してしまった場合は、手数料１０００円にてディスクをお取り替え致します。下記のものを巻末に記載しました“トンキンハウス「ユーザー・サポート係」”宛にご送付下さい。

・お問い合わせ用紙　　（p.295／必要事項をご記入下さい）
・手数料　１０００円　（切手、または現金書留にて）
・破損した製品ディスク

エンジニアズ・ハンドブック編

INTRODUCTION

新スタッフの諸君、ＯＳＩへようこそ。
諸君らも知っての通り ＯＳＩ(Organization of Strategic Intelli-gence：戦略情報機構)とは、永年にわたって国家防衛の計画立案にあたってきた国防省きっての頭脳集団である。従って新しいスタッフの採用に際しては大変厳しい審査が行なわれるが、これに見事パスした諸君に、まずはお祝いの言葉を述べておこう。
今回の審査は例年にも増して特に厳しいものであった。と言うのも今回の合格者は、或る極めて高度な任務に就くことが予定されているからである。
昨今、マスメディアを通じて耳にしているものもあると思うが、次期防衛構想における我が国地上軍の要は、現在のＳＲ-Ⅲ式からサイバータンクに取って替られることになる。
このサイバータンクとは、搭載するコンピュータに組みこまれたＡＩ（人工知能）の命令によって、無人で作戦行動を展開することのできる陸軍のいわば最終兵器である。１６年前、陸軍からの要請を受けた我がＯＳＩは、この兵器の開発を目指して極秘裡にひとつのプロジェクト・チームを発足させた。
コードネームを“オメガ”という。
そして先般、オメガ・プロジェクトのスタッフはＡＩプログラムの設計から戦闘シミュレーションの実行まで、サイバータンクの設計に関わるいっさいをオペレートするシステム・ソフト“OMEGA SYSTEM”の完成にこぎつけた。後はその潜在力をフルに活かした強力なＡＩの開発を待つばかりである。
諸君らの任務とは他でもない、このオメガ・プロジェクトの専属エンジニアとして勤務し、“史上最強のサイバータンク”を世に送り出すことなのである。

＊＊＊ エンジニアズ・ハンドブックの構成 ＊＊＊

PART 1

この章は、サイバータンクの設計に関する全ての基本的なノウハウについて記載する。

PART 2

この章は、メニュー内の全項目の機能について、モジュールごとに記載する。

PART 3

この章は、CCL コマンドの全てのヴァリエーションについて記載する。

PART 4

この章は、OSI カプセル・ルーチンの使い方について記載する。

PART 5

この章は、チーム・コンバットの設計方法について記載する。

PART 6

この章は、プレー中に表示されうる全てのエラー・メッセージについて記載する。

PART 1

この章は、サイバータンクの設計に関する全ての基本的なノウハウについて記載する。

SECTION 1
オリエンテーション

SECTION BRIEF
オメガ・プロジェクトの沿革、及び目的については INTRODUCTION に簡単に触れておいた。ここでは、サイバータンクの設計に関する具体的な研修に入る前に、諸君が、エンジニアとして知っておかなければならない最低限の知識、OMEGA SYSTEM の内部構造、全体の作業工程などについて解説する。

1．1　サイバータンク・エンジニアの昇級制度

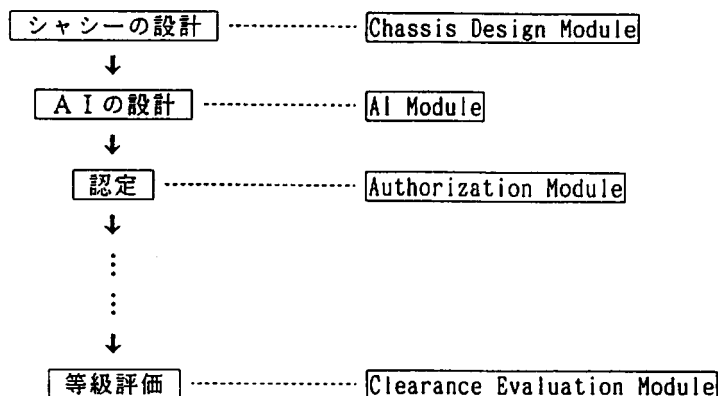
既に述べたとおり、陸軍の次期主力兵器の開発を任務とするオメガ・プロジェクトでは、エンジニアの質的向上と維持を図るために、１０等級にわたる昇級制度を採用している。
昇級を果たすためには、諸君が設計したサイバータンクを“等級評価”(Clearance Evaluation)と呼ばれるコンピュータ・シミュレーション(模擬戦闘)にかけ、コンピュータが用意した敵のタンクを相手に一定の成績(１０戦中７勝以上)を修めなければならない。
なお、エンジニアはその等級（Security Clearance）に応じて開発のための予算が割り当てられており、各エンジニアとも入門時は

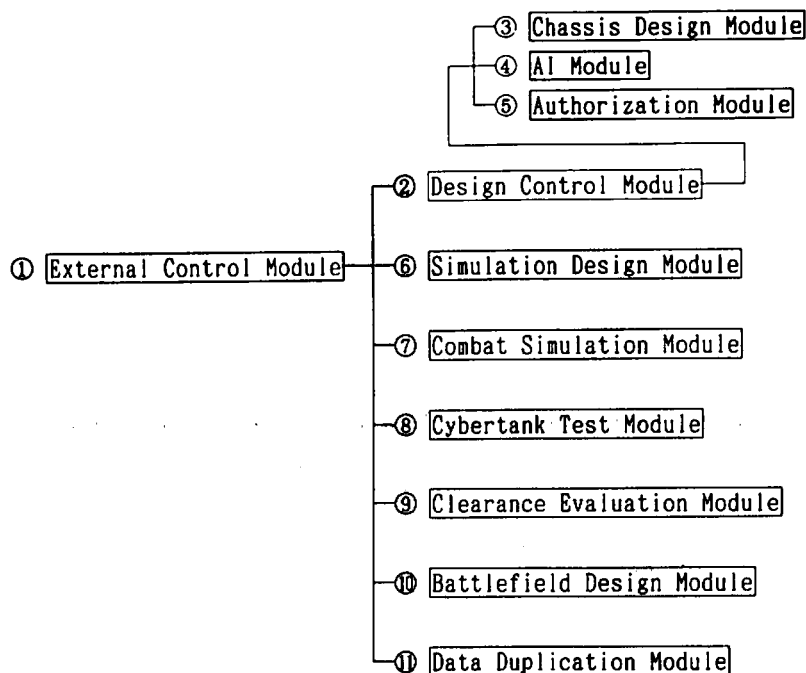
Security Clearance	: STANDARD
予算	: 1000 Credits

で、１ランク昇級するごとに予算が 1000 Credits ずつ加算される。
では次に、等級評価を受けるまでの作業工程について見てみよう。

1.2 作業工程とその処理について――モジュールの役割

諸君は今後、サイバータンクの設計に始まって等級評価に至る一連の作業をこのハンドブックに沿って進めていくことになるが、その過程には、タンクの設計とその評価という基本的な要素以外にもさまざまな工程が含まれる。その全容は p.32 の作業工程図に示した通りだが、ここではまず OMEGA SYSTEM がこれらの工程をどのような仕組みで処理しているのかその点について述べておこう。

OMEGA SYSTEM とは、INTRODUCTION にも触れた通り"ＯＭＥＧＡ"の全工程をオペレートするために開発されたシステム・ソフトであるが、内部はある特定の作業工程を分担して受け持つところの"モジュール"（Module）と呼ばれる１１のセクターに分かれて構成されている。
即ち、"ＯＭＥＧＡ"の各作業工程は１１のモジュールがこれを分業して処理に当たっているのである。
下の図は、例として工程の一部とそれを分担するモジュールの関係を表わしたものである。

1.3 モジュールの種類とそのアクセス

下の図は、OMEGA SYSTEM を構成する１１のモジュールの種類とそのアクセス系統を示したものである。

解説

①：External Control Module は、各モジュールにアクセスするための起点となる場所で、OMEGA SYSTEM の玄関口に当たる。

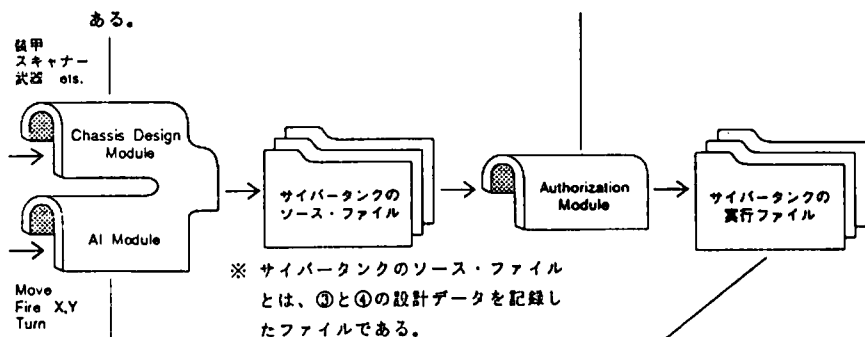
②：Design Control Module は③〜⑤のモジュールへの中継点である。

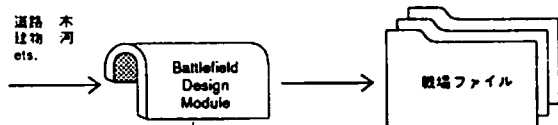
※　③〜⑪のモジュールの役割に関する解説は次頁を参照のこと

1.4　作業工程図

③：Chassis Design Module とは既製のコンポーネントを組み合わせて、サイバータンクの車両本体を設計するところである。

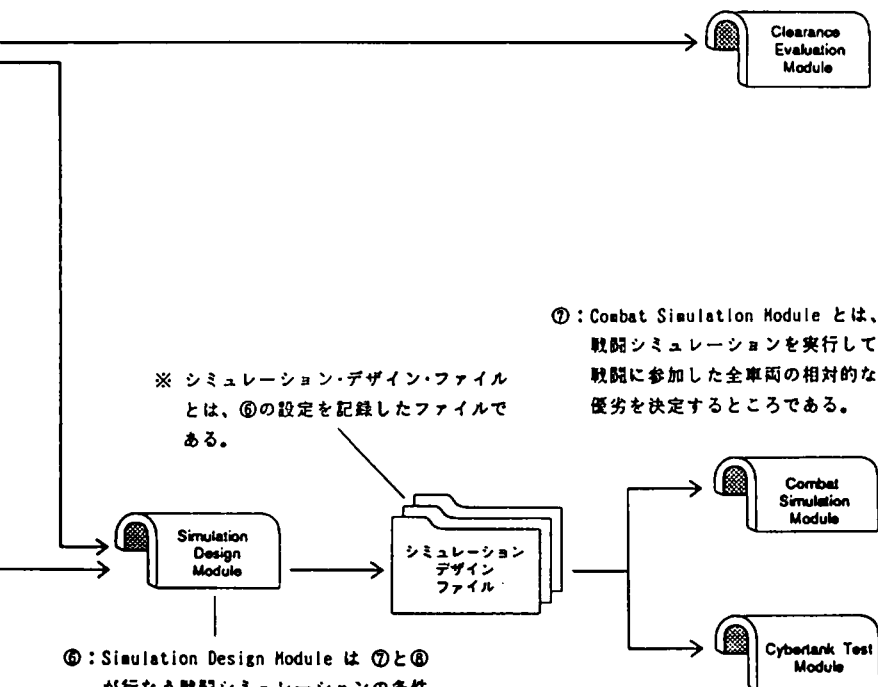
⑤：Cybertank Authorization Module はサイバータンクのソース・ファイルをコンピュータが直接処理することのできるマシン語に翻訳するところである。

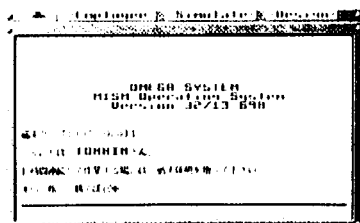
※ サイバータンクのソース・ファイルとは、③と④の設計データを記録したファイルである。

④：AI Module は、サイバータンクの頭脳にあたるＡＩ（人工知脳）をプログラミングするところである。

※ サイバータンクの実行ファイルとはソース・ファイルを⑤でマシン語に変換したファイルである。

⑩：Battlefield Design Module とはパーツを組み合わせて戦場マップを作成するところである。

※ 戦場ファイルとは、⑩で作成したデータを記録したファイルである。

⑪：Data Duplication Module はデータ・ファイルをコピーするところである。（図にはなし）

⑨：Clearance Evaluation Module は
等級評価を行なうところである。
Clearance Evaluation Module
⑦：Combat Simulation Module とは、
戦闘シミュレーションを実行して
戦闘に参加した全車両の相対的な
優劣を決定するところである。
※ シミュレーション・デザイン・ファイル
とは、⑥の設定を記録したファイルで
ある。
Combat Simulation Module
Simulation Design Module
シミュレーション
デザイン
ファイル
Cybertank Test Module
⑥：Simulation Design Module は ⑦と⑧
が行なう戦闘シミュレーションの条件
1) 自車両 2) 敵車両 3) 戦場
を設定するところである。具体的には
1)と 2)を実行ファイルの中から、3)を
戦場ファイルの中からそれぞれ選ぶ。
⑧：Cybertank Test Module とは、
主にＡＩプログラムのデバッグ
を行なうところである。

1.5 External Control Module (ECM)

★ External Control Module（以下 ECM）は各モジュールにアクセスするための OMEGA SYSTEM の起点にあたるが、ここには常時 OSI 本部からスタッフへの連絡事項や世界の最新ニュースが流されている。エンジニア諸君は常時これに目を通すよう心掛けること。

External Control Module

★ ECM 以下各モジュールでは、メニュー [≡] 内の各項目を選択することによって、（キーボード／マウス），（アナログ／液晶）の設定を変更することができる。

（キーボードの場合）[ESC] キーを押して [≡] を開き、[↑],[↓]キーで☜を目的の項目に降ろして[⏎]キーを押す。

（マウスの場合）[≡] をクリックしたまま下に引っ張り、目的の項目が反転したところで、クリック・ボタンを離す。

[Analogue] …… これを選択すると [Analogue] / [L.C.D.]（液晶）モードが交互に切り替わる。

[Keyboard] …… これを選択すると、キーボード入力モードになる。これは、[CTRL] +K で代用することができる。

[Mouse] ………… これを選択すると、マウス入力モードになる。これは、[CTRL] +L で代用することができる。

※ マウスの用意がないときに、間違って [Mouse] を選択してしまった場合は、[CTRL] +K でキーボード・モードに戻す。

SECTION 2
“ＡＬＰＨＡ”の設計

SECTION BRIEF

サイバータンクの設計には以下３つの作業工程がある。

① シャシー（Chassis）の設計
サイバータンクの車両本体を設計することで、既製のコンポーネントの中から、装甲、スキャナー、攻撃砲などを選び出し、これらを組み合わせることによって作成する。

② ＡＩ（Artificial Intelligence：人工知能）の設計
ＡＩとは、タンク搭載のコンピュータに組み込まれたタンクの行動を制御するためのプログラムを指し、CCL というプログラム言語でこれを作成する。

※ ①と②の情報が書き込まれた設計図を、サイバータンクの“ソース・ファイル”と呼ぶ。（p.32 参照）

③ 認定（Authorization）
CCL という高級言語で書かれた“ソース・ファイル”をタンクに搭載されたコンピュータが直接処理することのできるマシン語に変換する。この際、①，②の設計にミスや洩れがあれば、コンピュータがこれを指摘する。

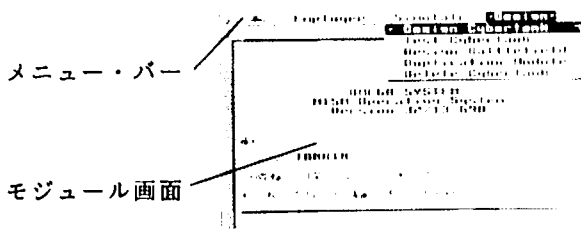
※“ソース・ファイル”の内容をマシン語に変換したデータをサイバータンクの“実行ファイル”と呼ぶ。（p.32 参照）

では、実際に１台のサイバータンクを設計しながら、各工程について詳しく見ていこう。

操作

★ キーボードを使用している場合、External Control Module 及び Design Control Module に入った直後は、☜が画面上に現われないので、これを表示するときは [ESC] キーを押す。

★ “メニューの [Design] を開いて [Design Cybertank] を選択”という指示のときは

ECM

（キーボードの場合）[←],[→]キーで、☜をメニューの [Design] に移動させ、次に[↑],[↓]キーで☜を下げ、[Design Cybertank] が黒く反転したところで[↵]キーを押す。
（マウスの場合）メニューの [Design] をクリックしたまま☜を下に引っぱり、[Design Cybertank] が黒く反転したところで指を離す。

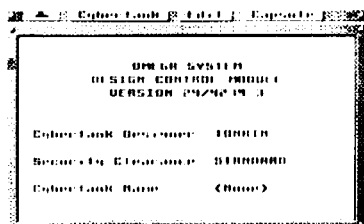
★ メニュー内の項目で、文字が灰色で表示されているものは、現状況下では選択できない項目であることを示している。

★ キーボード使用時、☜をモジュール画面からメニュー・バーに戻すときは [ESC] キーを押す。

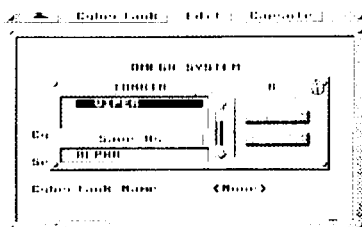
※ この他の操作方法に関しては、随時その箇所で解説する。

2．1　サイバータンクの登録手順

まず初めに我々は、これから作るサイバータンクの名前を OSI に登録しなければならない。以下、指示に従って実際に作業を進めていこう。

1) ECM に入ったら、メニューの Design を開いて Design Cybertank を選択し、Design Control Module に入る。(左頁の 操作 を参照)

Design Control Module

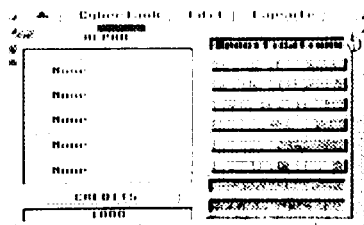
2) メニューの Cybertank を開いて、New を選択する。

3) サイバータンクの命名は自由であるが、PART 1 はこのマニュアルに沿って作業を進めるので、ここでは“ＡＬＰＨＡ”と入力する。

※“ＶＩＰＥＲ”は OSI 既製のサイバータンクで、あらかじめ諸君のＩＤディスクにコピーしておいたファイルである。

4) (キーボードの場合) ↑,↓キーで☞を Save に合わせて↵キー。
(マウスの場合) Save をクリックするか、↵キーを押す。
これで登録完了。画面は Chassis Design Module に転送される。

2.2　シャシーの設計

Chassis Design Module の画面を見ると、Specifications（仕様）の欄に

Tank Class
Fuel Cells
Drive System
Weapon Type
Scanner
Special Items

Chassis Design Module

の６つの項目がある。これらはサイバータンクの車両本体を構成する部位の名称で、シャシーの設計はこれらの部位に対して OSI があらかじめ用意した複数のコンポーネントの中から各々ひとつを選び出し、これを組み合わせるという方式を採る。
“ＡＬＰＨＡ”のシャシーの設計に入る前に、ここで各部位の機能とコンポーネントを選ぶときの決まりについて触れておこう。

2.2.1　各部位の機能

① Tank Class　　車体の骨組みにあたる部分で車体の重量、装甲の厚さなどを決定する。種類によっては、水陸両用の仕様になっているものがある。必須装備。

② Fuel Cells　　燃料。車両の移動のみならず、攻撃、探査など、サイバータンクが持つあらゆる機能のエネルギー供給源である。必須装備。

③ Drive System　　発電装置。コストに応じて発電容量が増え、車体の移動速度が上がる。必須装備。

④ Weapon Type　攻撃砲。種類によって破壊力と各部位の損傷効果が異なる。必須装備。

⑤ Scanner　探査装置。サイバータンクの目となり耳となる部分。タンクを中心とした扇形のエリアをスキャンし、敵の位置を突き止める。コストに応じて探査範囲の半径(単位：hm = 100m)と中心角(単位：°)が増す。必須装備。

⑥ Special Items　オプション。①～⑤の標準コンポにはない特殊な機能を発揮することができる。但し、そのうちの幾つかは、これ機能させるための特別なプログラムをＡＩの中に組み込まなければならない。

> ※ 各コンポのグレード別の細かな仕様、及び Special Items の仕様に関しては、P.42 ～ 45 を参照のこと。
> ※ 但し、Special Items の使用に関するプログラミングは、PART 3 で扱う高度な内容なので、ここではその種類と機能についてだけ押えておけばよい。

２．２．２ コンポーネントの選択におけるきまり

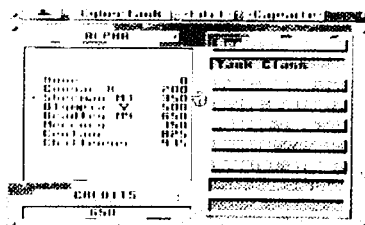
★ コンポーネントはグレードによって各々コストが決っており、各エンジニアは、その Security Clearance に応じて割当てられた予算の枠内でコンポーネントの選択を行なわなければならない。各エンジニアとも入門時の Security Clearance は “STANDARD” で、与えられる予算は 1,000 Credits である。

★ 各部位の機能でも触れたように、①～⑤に関しては必須装備で、かつその性質上、複数のコンポーネントを同時に装着することはできない。

★ Special Items はオプションで、予算の許す限りいくつでも同時に装着することができる。

では、“ＡＬＰＨＡ” のシャシーの設計に入ろう。

2.2.3 シャシーの設計

1) (キーボードの場合)[↑],[↓]キーで☜を画面右の [Tank Class] に合わせ [↵]キーを押す。左のウィンドに Tank Class のコンポーネントの一覧が出たら、[←],[→]キーで☜をウィンド内に移し、[↑],[↓]キーで☜を Sherman M7 に合わせて[↵]キーを押す。
(マウスの場合)画面右の [Tank Class] をクリックし、左のウィンドに Tank Class のコンポーネントの一覧が出たら、Sherman M7 をクリックする。

★ コンポーネントは選択されると左端の円内が赤に変わる。
★ [CREDITS] の欄は予算残高を示しており、コンポーネントが選択されると、自動的にその金額が差し引かれる。
★ 現予算を超過するコンポーネントは灰色で表示される。

2) 同様にして以下のコンポーネントを選択する。

[Fuel Cells] ………… 300 units
[Drive System] ……… Light
[Weapon Type] ……… Explosive
[Scanner] ……………… 20 Hm - 45°

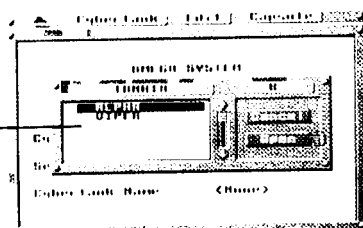
選択が指示通りになされていれば、この時点で [CREDITS] の残高は "0" になる。

3) 選択を終えたら、[Specifications] を選択する。画面左にいま選択したシャシーの一覧が表示されるので、指示通りの仕様になっているか再度確認する。

確認を終えたところで、次はいよいよＡＩの設計に入る。

＊＊＊ソース・ファイルのセーブとロード＊＊＊

※ ソース・ファイルのセーブは Cybertank の Save を選択するか、或いは Cybertank の Exit を選択して "Save changes to ALPHA ?" という表示が出た時点で Yes を選択する。
なお、作業を終了する場合は ECM に戻ってから、Employee の Call it a day を選択して OMEGA SYSTEM を閉じる。

※ ソース・ファイルのロードは、ECM に入ってから、Design の Design Cybertank を選び、Design Control Module に入る。
次に Cybertank の Load を選択して、ファイル・ウィンド(下図参照)を開き、目的のファイルを反転させてから Open を選択する。

(キーボードの場合) ←,→キーで☞をファイル・ウィンドに移し、↑,↓キーでロードするファイルを反転させてから、☞を Open に置いて↵キーを押す。
(マウスの場合) ロードするファイルをクリックしてこれを反転させてから、Open をクリックする。
この際、画面は直接 AI Module に転送されるので、シャシーの設計から始める場合は Cybertank の Chassis を選択する。

＊＊＊コンポーネントの仕様一覧＊＊＊

★ Tank Class	重量	装甲	水陸両用	価格
Cougar X	中	弱	不可	200
Sherman M7	重	弱	可	350
Olympia V	軽	弱	不可	500
Bradley M4	中	普通	不可	650
Mercury	中	弱	可	750
Centaur	軽	普通	不可	825
Challenger	重	強	不可	975
M5 Turtle	中	普通	可	1100
Britannia	中	強	可	1500
Bentley	軽	強	可	2000

※ 軽量のタンクほど移動速度が速い。

※ 水陸両用でないタンクが水に入ると、特に電気系統に著しい損傷を受ける。

★ Drive System	重量	速度レベル	価格
Light	軽	1	150
Standard	中	2	275
Heavy	重	3	400
Turbo	中	3	525
Dual-Turbo	軽	3	700
Gyro	重	4	900
Flux	中	4	1200
Fission	軽	4	1500
Fusion	中	5	2000
Ion	軽	5	2500

※ 速度レベルは数が大きくなるにしたがって速くなる。

※ Drive System の重量が重いと、移動速度が鈍り、燃費も悪くなる。

★ Weapon Type	連射速度	破壊力	価格
Piercing	遅い	小	175
Explosive	遅い	小ー中	250
HEAT	遅い	中	475
Laser	速い	中	675
Turbo Laser	速い	中ー大	750
Gauss Gun	普通	大	900
Plasma Gun	普通	大	1300
Nuke	遅い	大	1800

※ 弾頭を使用するものは、装填に時間を要するために連射速度が遅い。

※ 各部位に対する損傷効果は兵器によって異なる。

※ HEAT : High Explosive Anti-Tank

＊＊＊ Special Items の仕様一覧 ＊＊＊

Special Items は、これを装着しただけで機能するものと、機能させるために特別なプログラムを要するものとの２つに分かれる。
プログラムの組みかたについては、それぞれの参照頁に記載した。
なお、Special Items は予算内であれば幾つでも同時に装着することができる。

Energy Miser　燃料消費を約５０％節減する。プログラム不要。

Comm-Link　チーム・コンバット(チーム対チームの対戦)のとき味方チーム員との通信手段として使用される。これを装着していないタンクはチーム員との交信ができない。プログラミングに関しては p.248 参照。

Repair Kit　多目的修理用キット。サイバータンクの全ての部位の修理に使うことができる。１キットにつき４回の使用が可能。プログラミングに関しては p.211 参照。

Scanner Lock　発見した敵車両に対してこれを使用すると、あとはスキャナーが自動的にその敵を捕捉し続ける。捕捉した車両が破壊されるか、もしくはスキャナーの域外に逃げた時点で、ロックは自動解除される。任意に解除することも可能。プログラミングに関しては p.204 参照。

Listener　これを装備していると、敵サイバータンクが自車両をスキャナー・ロックしているかどうかをチェックすることができる。
プログラミングに関しては p.205 参照。

Jammer	敵サイバータンクによってスキャナー・ロックされたとき、妨害電波を出してこれを解除する。 Jammer は Listener を設置しなくても機能する。 プログラミングに関しては p.206 参照。
Launcher	リモート・スキャナーを空中に発射し、最も近くにいる敵サイバータンクを探知する。 Launcher 1基に4発のリモート・スキャナーが装填されている。 プログラミングに関しては p.207 参照。
Shield	シールドを上げることによって、敵サイバータンクからの攻撃によるダメージを最小限に抑えることができる。シールドの上げ下げはプログラムによって任意に行なうことができるが、シールドを上げているときは次のような他システムへの影響がある。 1)スキャナーの探査域が半減する。 2)消費燃料が悪化する。 3)攻撃不能になる。 プログラミングに関しては p.213 参照。
Accelerator	サイバータンクが搭載するコンピューターの処理速度を上げ、ロジック・コマンド（p.130）の処理時間を1／2に短縮する。但し、アクション・コマンド（p.128）の処理速度は変らない。 プログラミング不要。

2.3　AIの設計

AIの設計とは、OSI が独自開発した、Cybertank Command Language (CCL)を使って、サイバータンクの作戦行動をプログラミングすることである。

2.3.1 Cybertank Command Language (CCL) とは

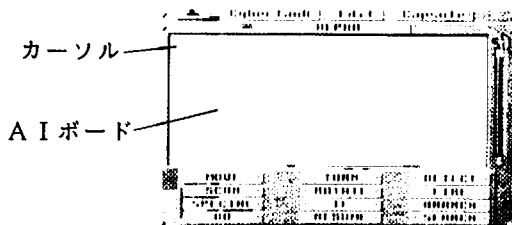
この Cybertank Command Language（以下 CCL と略す）とは、タンクに搭載されたコンピュータが処理することのできる唯一のプログラム言語であり､サイバータンクの行動は全てこの CCL を使って書かれた"コマンド"（サイバータンクに具体的なひとつの行動や判断をさせる1センテンスの命令）によって制御されている。従って優れたサイバータンク･エンジニアになるには、この CCL に精通することが絶対の条件であることを忘れてはならない。下に、実際のプログラムに使われるコマンドのいくつかを挙げてみたのでちょっと見てみよう。

Scan for enemy tank	(敵戦車を探知せよ)
Move tank forward 1	(1hm 前進せよ)
Fire weapon at enemy tank	(敵戦車を砲撃せよ)

ご覧の通り、CCL は英語をベースにして開発された高級言語であることが判っていただけたと思う。従って、個々のコマンドの内容は初級エンジニアの諸君にも容易に理解できるようになっている。
しかし、当然のことながらプログラム言語である以上、使える単語や構文、コマンドの配列方法などには数多くの約束ごとがあり、これらのことを全て踏まえたうえで1行1行プログラムを組んでいくことは今の段階の諸君にとって容易なことではない。そこで初めての経験である"ALPHA"のAIの作成に関しては、我々は"セミ・カスタムデザイン"という大変便利な方法を採用したいと思う。

2.3.2 セミ・カスタムデザインによるAIの作成

セミ・カスタムデザインの解説は後にして、まず以下の手順に従ってAIを入力しよう。

1) メニューの Cybertank を開いて AI を選択し、AI Moduleに入る。（シャシーの設計に戻るときは Chassis を選択する）

AI Module

2) AIボードの上、左端にカーソルがあるのを確かめたら、まず下の注意事項を読み、それから以下のコマンドをキーボードで入力する。

Start ⏎	（ラベル名）
Do Seek ⏎	（ルーチン "Seek" を実行せよ）
Do Destroy ⏎	（ルーチン "Destroy" を実行せよ）
Branch to Start ⏎	（ラベル "Start" に分岐せよ）

注意事項

- 入力は大文字と小文字のどちらを使ってもかまわない。（OMEGA SYSTEM は大文字と小文字の識別をしない）
- "Start" は一番上の行の左端から打ちこむ。
- 改行は全て、⏎キーで行なう。この際、カーソルは自動的にインデント（左端から数文字さがったところに移動）されるので、"Do Seek" 以下のコマンドはその場所から打ちこむ。

解説
このＡＩはスキャナーで敵を探査追尾して砲撃、破壊したら初めに戻ってこれを繰り返すという内容のものである。上から順に解説しよう。

Start　これは“ラベル”と呼ばれるもので、Do Seek から Branch to Start までのルーチンを指す名札のようなものと理解すればよい。なお“ルーチン”とはサイバータンクにある特定の任務を実行させるCCLコマンドのつながりで、ＡＩの一部もしくは全体を成すものである。

Do Seek　これは“Seek”という名前の“カプセル・ルーチン”を実行せよというコマンドである。この“カプセル・ルーチン”とは、OMEGA SYSTEM が持つカプセル・ライブラリーに登録されている、或る特定の任務を実行する OSI 既製のルーチンである。

Do Destroy　“Do Seek”同様、“Destroy”という名前で登録されているカプセル・ルーチンを実行せよというコマンドである。

Branch to Start　“Branch to *** ”とは、“ラベル *** に飛び、そこから処理を再開せよ”という意味のコマンドで、“*** に分岐せよ”と訳す。コマンドの処理は、特別の指示がない限り上から順番に行なわれるが、このコマンドによって“ＡＬＰＨＡ”のＡＩは“Destroy”ルーチン終了後、ラベル“Start”に戻って循環構造を形成する。

以上の４行で、ＡＩプログラムとしての体裁は整ったように見えるが、実はカプセル・ルーチンを利用する場合、特別に組み込まなければならないコマンドがある。
上にも述べたように、“Seek”“Destroy”というのは、カプセル・ルーチンのいわばコードネームであって、CCL コマンドの例にあげた“Scan”や“Move”のように、OMEGA SYSTEM が直接判読、実行のできる正式な CCL の単語ではない。従ってカプセル・ルーチンを使うときは、OMEGA SYSTEM にそれがライブラリーに登録されているカプセル・ルーチンであるということを、教えてやらなければならない。

これを実行してくれるのが

```
Include Seek          (ルーチン "Seek" を読み込め)
Include Destroy       (ルーチン "Destroy" を読み込め)
```

というコマンドである。このコマンドは具体的には、"Seek" 或いは "Destroy" というカプセル・ルーチンをライブラリーから捜し出してきて、それを "ＡＬＰＨＡ" のＡＩに取り込む作業を行なう。
この２行を挿入することによって、実際には２３行におよぶ "Seek" ルーチン（p.51 参照）と、７行の "Destroy" ルーチン（p.52 参照）が "ＡＬＰＨＡ" のＡＩに組み込まれるのである。

> ※ カプセル・ルーチンの使用方法に関しては、PART 4 で改めて詳しく解説する。

では、上の２行を "Branch to Start" の下に続けて入力し、これまでの作業が正しくなされているか下と照らし合わせてみよう。

```
Start
    Do Seek
    Do Destroy
    Branch to Start

    Include Seek
    Include Destroy
```

●レイアウト上、行を空けてもプログラム的な支障はない。

以上見てきた通り、セミ・カスタムデザインとは、ある特定の任務を果たすべく設計された既製のルーチン（例えば "Seek" は敵の探査と追跡が専門で、"Destroy" は攻撃が専門である）を必要に応じてうまく組み合わせ、インスタントにＡＩを組んでしまう方法を指す。

これに対し、プログラムを1行1行作成していくやり方をフル・カスタムデザインと呼ぶが、これに関しては SECTION 5 で詳しく解説する。
これで“ＡＬＰＨＡ”のソース・ファイルは一応完成した。次に我々は設計の最終段階であるサイバータンクの“認定”手続きに入る。

--- * * * ---

“Seek”のコマンド・リスト

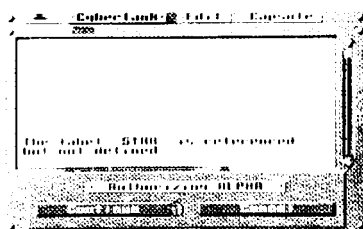
```
Seek
    Do Onward
    Rotate Scanner Left 1
Look
    Scan for enemy tank
    If enemy tank was not found then branch to Seek
    If enemy tank is beyond range then branch to Track
    Resume
Track
    Align tank with scanner
    Do Onward
    Branch to Look
Onward
    Detect obstruction at tank direction
    If ObstacleType = 0 then branch to Go
    Fire at tank direction
    Detect obstruction at tank direction
    If ObstacleType = 0 then branch to Go
    Turn tank left 3
    Branch to Onward
Go
    Move tank forward 3
    Resume
```

"Destroy" のコマンド・リスト

```
Destroy
    Fire at enemy tank
    Fire at enemy tank
    Fire at enemy tank
    Scan for enemy tank
    If enemy tank is within range then branch to Destroy
    Resume
```

2．4　サイバータンクの認定（Authorization）

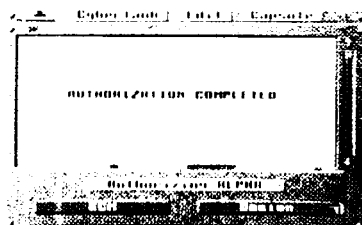
サイバータンクの認定(Authorization)とは、OMEGA SYSTEM が CCLで書かれたＡＩプログラム、及びシャシーの設定をコンピュータが直接処理することのできるマシン語に変換し、この際これと並行して設計におけるミスや漏れが無いかを洗い出すものである。
設計に誤りがある場合、OMEGA SYSTEM はエラー・メッセージを出して、具体的にその箇所と内容を指摘する。設計ミスの一般的なものとしては、シャシーに関しては装備の漏れ、ＡＩに関しては主にロジックの矛盾やスペル・ミスなどが挙げられる。
エンジニアはそれぞれのモジュールに戻って指摘された誤りを修正し、再び認定を受け、パスするまでこれを繰り返さなければならない。
こうして認定をパスしたサイバータンクだけが OSI に正式登録され、次の段階である“戦闘シミュレーション”や“等級評価”に進む資格を得るのである。
では、メニュー Cybertank の Authorize を選択して、“ＡＬＰＨＡ”の認定作業に入ろう。

★ 認定に失敗した場合

Cybertank Authorization Module

誤りがあると図のように、画面下にエラー・メッセージが現われる。
ここで Continue を選択すると、他にも設計ミスのある場合はこれを指摘する別のエラー・メッセージが追加表示される。
Cancel を選択すると画面は誤りのあったモジュールに転送される。

例として挙げた画面のエラー・メッセージは、“Branch to Start” を “Branch to Star” と打ち間違えたときのものである。直訳すると「ラベル “Star” が言及されているが、定義されていない」となる。これは即ち「ラベル “Star” に行けと言うが、“Star” なんていうラベルはどこにも見当らないぞ」という意味である。

> ※ エラー・メッセージは、その原因に対応してきわめて多様なものが用意されている。
> 個々のエラー・メッセージに関する解説は PART 6 を参照。

★ 認定に成功した場合

この表示が出れば認定完了である。
[ECM] を選択すると、OMEGA SYSTEM が “Save changes to ALPHA ?” と訊いてくるので [Yes] を選択する。
[Design] を選択すると AI Module に戻る。

以上で、Authorization Module の機能は理解して頂けたと思う。
では、こうしてマシン語に変換されたＡＩプログラムのデータはどこのファイルに保管されるのだろうか?
ここでサイバータンクのファイル管理について１度整理しておこう。

2.5 サイバータンクのファイル管理について

サイバータンクの設計内容を記録したファイルには、データの持ち方によって“ソース・ファイル”と“実行ファイル”の２種類があることは既にお話した通りである。
ここで、これらのファイルの開設とセーブのシステムについて整理しておこう。

● ソース·ファイル

ソース·ファイルの開設は Design Control Module でエンジニアが新規にサイバータンクの名前を登録（セーブ）した時点で行なわれ、作成したＡＩプログラムや選択したシャシーのデータは全てここに記録される。
セーブは Design Control Module のメニュー [Cybertank] の [Save] を選択して行なう。
但し、OMEGA SYSTEM はエンジニアがいちいちこの操作を行なわなくても済むように、またこれを忘れてせっかく作成したプログラムを失ってしまうことがないように、新規にサイバータンクを開設した場合や、既存のファイルに何らかの修正を加えた場合は、Design Control Module を出る際、必ず “Save changes to *** ?” という確認の質問を出すように設計されている。従って、エンジニア諸君は、ここで単に [Yes] を選択すればよいのである。

● 実行ファイル

実行ファイルは、ソース·ファイルが Authorize に成功した時点で自動的に開設される。即ち、Cybertank Authorization Module はソース・ファイルをマシン語に変換すると同時に、そのデータを記録したファイル――実行ファイルを作成し、これをセーブするという機能を持っている。
なお、実行ファイルのファイル名はソース・ファイルと同じものが自動的に付けられる。

さて、諸君が初めて設計したサイバータンク“ＡＬＰＨＡ”が実戦でどんなパフォーマンスを見せてくれるか。次はこれをシミュレーションしてくれるモジュールの解説に入る。

SECTION 3
戦闘シミュレーション

SECTION BRIEF

認定に成功した実行ファイルは p.32, 33 の“作業工程図”にある通り、直接 Clearance Evaluation Module に進んで等級評価を受けるか、またはシミュレーション・デザイン・ファイルの作成工程を経て Combat Simulation Module や Cybertank Test Module に進み、ここでタンクの性能を前もって確かめることができる。
この3つのモジュールはそれぞれ別個の機能と用途を持っているが、サイバータンクの実行ファイルをロードして、コンピュータ上で模擬戦闘を行なうという点では共通しており、これらが処理する模擬戦闘を“戦闘シミュレーション”と呼ぶ。

Clearance Evaluation Module は、与えられた戦場で、与えられた敵を相手に戦闘を行なうものであるが、 Combat Simulation Module と Cybertank Test Module では戦場と敵車両の指定をエンジニアが自由にこれを設定することができる。
この設定を行なうのが Simulation Design Module であり、その設定条件を記録したものが“シミュレーション・デザイン・ファイル”である。
Combat Simulation Module と Cybertank Test Module は前者が戦績評価用ツールとして、後者がＡＩプログラムのデバッグ用ツールとして主に用いられるが、Cybertank Test Module の使用はフル・カスタム方式によるプログラミングの知識を前提とするので、SECTION 7 に譲り、この節では Simulation Design Module と Combat Simulation Module の使い方について記述する。

では、シミュレーション・デザイン・ファイルの作成から始めよう。

3.1 シミュレーション・デザイン・ファイルの作成

“シミュレーション・デザイン・ファイル”とは、Combat Simulation Module と Cybertank Test Module が行なう戦闘シミュレーションの設定条件を記録したファイルのことで、その条件とは次の3つを指す。

1) 分析の対象となる車両（自車両）
2) 敵車両
3) 戦場

シミュレーション・デザイン・ファイルの作成とは 1)と 2)をサイバータンクの実行ファイルの中から、3)を戦場ファイルの中からそれぞれ選択するという作業を指す。その選択に際しては以下の決まりがある。

★ 1つのシミュレーション・デザイン・ファイルには、最低で2両（自車両と敵1両）、最高で15両（自車両と敵14両）の戦車を登録することができる。
★ 同一車両を複数登録することができ、自車両に選んだタンクを、同時に敵車両として登録することもできる。

では、次の[操作]を参考にしながら作業に入ろう。

操作

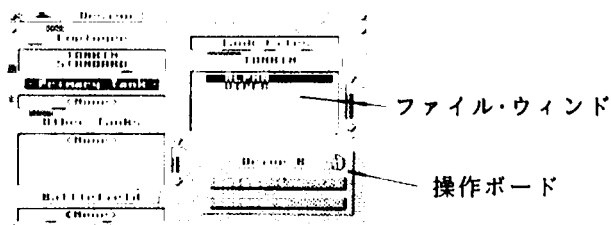
★ 項目 [Primary Tank], [Other Tanks], [Battlefield] の選択
(キーボード) [←],[→]キーで☞を操作ボード(右図参照)に移し、更に、[↑],[↓] キーで [Category] に下ろして[↵]キーを押す。
(マウス) 項目を直接クリックする。

★ ファイルの選択
(キーボード) [←],[→]キーで☞をファイル・ウィンドド内に移し、[↑],[↓]キーで選択するファイルを反転させ、[↵]キーを押す。
(マウス) ファイルをダブル・クリックする。

★ Other Tanks からのファイルの削除

（キーボード）☜を Other Tanks のウィンドウ内に移してから ⇧,⇩ キーで削除するファイルを反転させ、⏎キーを押す。

（マウス）Other Tanks のウィンドウ内で、削除するファイルをダブルクリックする。

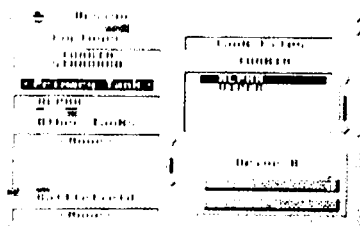
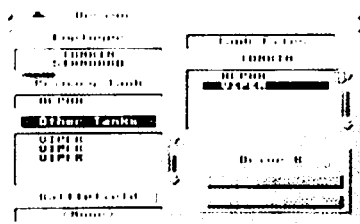
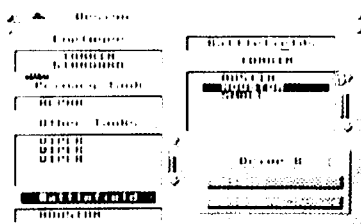
1) メニューの Simulate を開いて Design a Simulation を選択し、Simulation Design Module に入る。

Simulation Design Module

★ 画面左の４つの項目はそれぞれ以下の内容を表わす。

Employee ……………… エンジニアの登録名と階級
Primary Tank ………… 分析の対象となる車両（自車両）
Other Tanks …………… 敵車両
Battlefield ………… 戦場

★ 黒く反転した項目は、入力待ちの状態であることを表わし、ファイル・ウィンドには、その項目のファイルの一覧が表示される。

画面はいま Primary Tank の項が反転しており、ウィンド内には“ＡＬＰＨＡ”と“ＶＩＰＥＲ”が表示されている。これで諸君の設計した“ＡＬＰＨＡ”が確かに、OSI に正式登録されていることが判る。

2) Primary Tank を反転させ、ファイル・ウィンドド内の“ALPHA”を選択する。

3) Other Tanks を反転させ、“VIPER”を3回選択する。

4) Battlefield を反転させ、“HOUSTON”を選択する。

※ ファイル・ウィンドド内の“AUSTIN”,“HOUSTON”,“SMALL”は、OMEGA SYSTEM があらかじめ諸君のIDディスクに転送しておいた OSI 既製の戦場ファイルである。

※ OMEGA SYSTEM は、オリジナルの Battlefield を作成する Design Battlefield Module を備えており、この節の最後に、その使用方法に関する解説が記載されている。
このモジュールが持つ１つの重要な役割は、戦場に必要な要素だけを盛り込むことによって、より明確に、かつ効率よくタンクの仕様を検証することにあり、サイバータンクが持つあらゆる機能の特性を見極めるには、エンジニアにとって無くてはならない機能と言うことができる。
但し、PART１の研修では実際にこれを用いる場面はないので、いまはこれを読み飛ばし、諸君が実際にこれを必要とする段階が来たところで、読み返していただきたい。

5) 以上３項目の選択を終えたところで、このシミュレーション・デザイン・ファイルに名前を付けてセーブする。
メニューの Design を開いて、Save Simulation Design を選択し、"ＡＬＰＨＡＳＩＭ" とキーボードから入力して Save を選択する。
なお、既存のファイルを修正してこれを元のファイルにセーブするときは、以下の操作に従う。
（キーボードの場合）☞をファイル・ウィンドウ内に移して↑,↓キーで元のファイル名を反転させ、⏎キーを押してから Save を選ぶ。
（マウスの場合）ファイル・ウィンドウ内の元のファイル名をダブル・クリックしてから Save をクリックする。

これで、シミュレーション・デザイン・ファイルの作成は完了である。

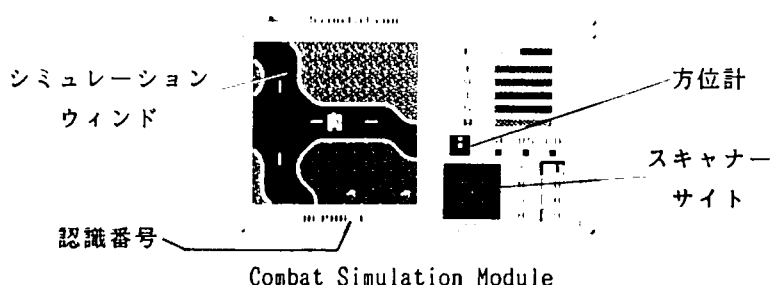
※ Simulation Design Module には、複数対複数の車両が戦闘を行なう "チーム・コンバット" 用のシミュレーション・デザイン機能がある。詳しい解説は PART 5 参照。

3.2 戦闘シミュレーションの実行

では、Combat Simulation Module を使って戦闘シミュレーションがどんなものか実際に見てみよう。

1) メニューの Simulate を開いて Start a Simulation を選択し、ファイル・ウインドが開いたら"ALPHASIM"を反転させ、最後に Open を選ぶ。
   なお、"ORIENTAT"は"VIPER"を Primary Tank とする OSI 既製のシミュレーション・デザイン・ファイルである。
2) 戦闘が始まったら決着がつくまでじっくりとこれを観戦する。
3) 決着がついたらメニュー Simulation の Restart を選択し、再び戦闘に入ったところで SPACE キーを押してポーズをかける。
   (ポーズ解除は再度 SPACE キー)

3.2.1 Combat Simulation Module の見方

Combat Simulation Module

解説

★ 戦闘シミュレーション上では、登録されている全ての車両が相互に攻撃し合い、最後に生き残ったものがそのシミュレーションの勝者となる。なお燃料切れの車両は破壊されたものとみなされる。

★ シミュレーション・ウィンドは常に、Primary Tank を中心とする縦横9×9hmのエリア(= Primary Tank の射程域)を映し出す。

★ 画面右上のインジケータに関しては以下の通りである。

F	Fuel Remaining ： 緑色の部分が燃料の残りを示す。燃料がきれるとタンクの全ての機能が停止する。
I	Internal Damage ： 電気系統の損傷。赤い部分が損傷の割合を示し、これが100%に達するとタンクは完全に破壊される。
A	Armor Damage ： 装甲の損傷。損傷が100%に達するとタンクは完全に破壊される。
T	Tread Damage ： トレッド（=キャタピラ）の損傷。損傷が100%に達するとタンクは走行不能になる。
S	Scanner Damage ： スキャナーの損傷。損傷が100%に達すると敵の探知が不能になる。
W	Weapon Damage ： 攻撃砲の損傷。損傷が100%に達すると攻撃不能になる。

★ 方位計内の赤い四角は車両の向いている方向を、青い四角はスキャナーの向いている方向をそれぞれ示し、ふたつが同一方向に重なった場合はピンク色になる。

★ インジケータ下のＳＬ，ＤＳ，ＬＤ は、特定の Special Items (p.44 参照)を装着しているときに機能する計器である。

ＳＬ	敵車両にスキャナーをロックしたとき点灯する。“Scanner Lock” 装着時のみ。
ＤＳ	防御用シールドを上げているとき点灯する。“Shield” 装着時のみ。
ＬＤ	敵車両によって自車両がスキャナーロックされているとき点灯する。“Listener” 装着時のみ。

★ 敵を探知すると、スキャナー・サイト上に敵の位置が白いブリップで表示される。（ピンクの四角枠は射程域を示す）

★ 画面右下のカウンターに関しては以下の通りである。

T	戦闘シミュレーションの実行回数。
B	終了した戦闘の回数
S	勝利した戦闘の回数
A	現在生き残っている車両の数

※ 戦闘シミュレーションの実行回数は任意に設定することができる。操作はメニュー Simulation の Set Number of Battles を選び、（キーボードの場合）←,→キーでをカウント表示のところに合わせ、↑,↓キーで回数を設定し、最後に OK を選択する。（マウスの場合）左右の矢印をクリックするか、四角いレバーをクリックしたまま左右に引っぱり、最後に OK をクリックする。

★ Battlefield 全体を見渡すときは、メニューの Simulation を開いて Satellite View を選択する。この際、自車両は黄色いブリップ(機影)で、敵車両は白いブリップで表示される。この操作は CTRL +V で代用することもできる。解除は再度 CTRL +V。

★ “タンク・セレクションキー” とタンクの “認識番号”
戦闘シミュレーションでは “タンク・セレクションキー” を押すことによって Primary Tank に限らず、敵車両の行動やその計器盤を観察することができる。
OMEGA SYSTEM はシミュレーション・デザイン・ファイル作成の際、Primary Tank を “1”、Other Tanks に関してはそれが選択された順に従って “2” “3” …と全ての車両に “認識番号” を付けており、これと同じ番号のフル・キー上列の数字キーを押すことによって、その車両にシミュレーション・ウィンドのスポットをあてることができる。

この際、計器類の値もその車両のものに切り換わる。

例）認識番号“８”のタンクを見る場合 ……… “８”
　　認識番号“１０”のタンクを見る場合 ………（SHIFT +）“０”
　　認識番号“１３”のタンクを見る場合 ……… SHIFT + “３”

また（+），（−）キーによって、それぞれ前と後の番号のタンクにスイッチすることもできる。
この“タンク・セレクションキー”は次頁の Position Cybertank 機能や Cybertank Test Module でも使用するので覚えておくこと。

では、SPACE キーを押して戦闘を再開させ、以上の計器類が解説の通り機能していることを確かめてみよう。

終了するときは Simulation の Exit Simulation を選択する。

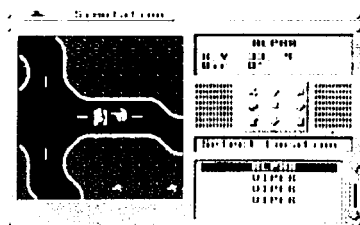
※ 以下３つのモジュールが行なう戦闘シミュレーションはサイバータンクの“実行ファイル”を読み込んでこれを処理している。

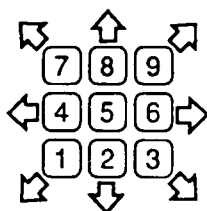
Clearance Evaluation Module
Combat Simulation Module
CybertankTest Module

従って、“ソース・ファイル”の内容を修正した場合はこれをセーブするだけでなく、改めてこれを Authorize することによって“実行ファイル”のデータも新しいものに書き換えてやらなければならない。これを忘れて戦闘シミュレーションを実行すると、古いデータのままの実行ファイルをロードすることになる。

3.2.2 "Position Cybertank" の使い方

戦闘を開いたとき、Battlefield における各タンクの位置はランダムで決定されるが、メニュー Simulation の Position Cybertanks を使うと、全てのタンクを任意の場所に配置し、また車両の向きを設定することもできる。

1) シミュレーションが始まったら Position Cybertanks を選択して下の画面に入る。

2) タンクの選択は、タンク・セレクションキーを押すか、或いは（キーボードの場合）ESC キーで☞をファイル・ウィンドに移して ↑, ↓ キーでファイルを反転させる。
（マウスの場合）ファイルを直接クリックする。
この時点で選択されたタンクが、シミュレーション・ウィンドの中央に映し出される。

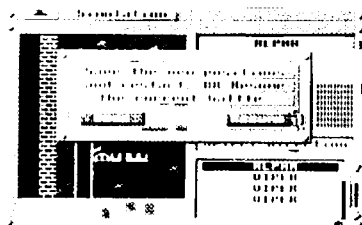
3) 位置の移動と車両の向きの設定は以下のようにして行なう。
（キーボードの場合）
・まず、ESC キーで☞をシミュレーション・ウィンド内に移す。
・テンキー（次頁参照）を使って、車両を希望の位置に移動させる。
・テンキーの"5"か、↵キーを押す。この時点で画面右中央の表示が Select Location から Select Direction に変わる。
・同じくテンキーを使って車両の向きを決める。

※ ノート版使用のときは NUM キーをロックして、テン・キーに代用されるフル・キーで操作する。

（マウスの場合）

・画面右の矢印をクリックして、車両を希望の位置に移動させる。

・矢印中央の四角をクリックするか、⏎キーを押すと、矢印の下の表示が Select Location から Select Direction に変わる。

・同じく矢印をクリックして、車両の向きを決める。

4) Simulation の Restart を選択して、下のウィンドを開く。

5) Save を選択すると、パラメータ類を全てリセットして、設定した位置から新規にシミュレーションを開始する。但し、この場合は車両の向きはリセットされてしまう。（即ち、direction "0" を向く）Resume を選択すると、設定した位置で設定した方角を向いた状態で、現在のシミュレーションを再開する。即ちインジケータの値はこのモードに入る前の値が継承される。

3.2.3 シミュレーション結果のプリント・アウト

OMEGA SYSTEM は、シミュレーション・ファイルに登録されている車両の過去の戦績を、全てシミュレーション・デザイン・ファイルごとに記録しており、エンジニアはその統計値を印刷することができる。
操作手順は以下の通りである。

1) まず ECM に戻って、Simulation の Print Simulation Stats. を選択する。
2) シミュレーション・デザイン・ファイルのウィンドが開いたら、プリント・アウトするファイルを反転させて、最後に Open を選択する。

以下はプリント・アウトされた書式のサンプルである。

```
--------------------------------------------------------------------
                          S T A T U S
--------------------------------------------------------------------

                 NUMBER OF TANKS    :  4
                 NUMBER OF BATTLES  : 12

                  [CUMULATIVE RANKINGS] (累積ランキング)

    TANK NAME          1   2   3   4   5   6   7 (位)

      ALPHA            1   3   7   1
      BETA             4   4   3   1
      GAMMA            6   4   0   2
      DINKY            1   1   2   8
```

解説

★ 表の第1行目に並ぶ1～7の数字は順位を表わし、その下に並んだ数字は各タンクがその順位を取った回数を表わす。順位は最後まで生き残ったタンクが1位、最後から2番目に生き残ったタンクが2位、という仕組みで決定する。

★ ランキングを受けるのは7位までである。従って8両以上で戦った場合、7位以下は全て同順として7位にカウントされる。

具体的に見てみよう。ここでは“GAMMA”が優勝回数6回と1番優秀な成績を修めており、それを優勝回数4回の“BETA”が追っている。また“ALPHA”と“DINKY”を比較してみると、優勝回数は1回で差がないものの、2位になった回数で“ALPHA”が勝っているので、相対的には“ALPHA”のほうが優れていると言うことができる。
このようにシミュレーション結果の統計値を取ることは、性能の優劣を判断する際、なるべく偶然の要素を捨象して、評価をより正当なものに近づけるという利点を持つのである。

※ OMEGA SYSTEM には、ディスク間のファイル·コピーを行なう Data Duplication Module が備わっており、これを利用することによって、諸君は自分が設計したタンクと他のエンジニアが設計したタンクを Combat Simulation Module 上で自由に戦わせることができる。
Combat Simulation Module 本来の機能はこのように、あらかじめ用意された敵と対戦する Clearance Evaluation Module と異なり、任意の車両同士を対戦させることができる点にある。ここに解説したプリント・アウト機能も、このような場合の戦績評価にこそ、その有効性を発揮するのである。
なお、Data Duplication Module の使い方に関しては p.165 の SECTION 9 を参照のこと。

3.3 Battlefield のオリジナル・デザイン

OMEGA SYSTEM には Battlefield を自由にデザインすることのできる Battlefield Design Module が備わっている。但し、このモジュールが持つ本来の役割は、戦場のヴァリエーションを増やすということよりも、むしろシャシーや開発中のＡＩの仕様を検証する際に Battlefield を必要な要素だけで構成することによって、目的に応じた理想的な実験環境を実現し、あらゆる角度からの分析を可能にすることにある。
Battlefield の作成は基本的には手作業であり、根気の要る仕事であるが、エンジニア諸君はこれをフルに活用することによって、各コンポーネントの性能や CCLコマンドの特性などに対する理解を更に深め、設計技術の向上に役立ててほしい。

さて、Battlefield の作成は、具体的には一定のエリア内に OSI既製のパーツを埋め込むことによって行なわれるが、このパーツの最小単位を“タイル”と呼ぶ。
ではまず、この“タイル”の使い方から解説しよう。

3.3.1 タイルの使用方法

ECM に戻ったら、メニュー [Design] の [Design Battlefield] を選択して、Battlefield Design Module に入る。

Battlefield Design Module

操作

(キーボードの場合)

★ ☜のメニュー・バー、エディット・ウィンド、及びパーツ表示エリアへの移動は、[ESC] キーで行なう。

★ エディット・ウィンド内での、Battlefield のスクロールは☜をエディット・ウィンドに合わせてから、[←],[→],[↑],[↓]キーによって行なう。

★ パーツ表示エリアのスクロールは、☜をパーツ表示エリアに合わせてから、[↑],[↓] キーによって行なう。

★ タイルの指定(選択)は、☜をパーツ表示エリアに置いてから、[←],[→],[↑],[↓] キーによって、黄色い四角枠を指定するタイルの上に合わせる。

(マウスの場合)

★ Battlefield のスクロール、及びタイル表示のスクロールはそれぞれのスクロール・バーによって行なう。

★ タイルの指定(選択)は、指定するタイルをクリックして、そのタイルの上に黄色い四角枠を合わせる。

タイルの大きさはちょうど1セル (p.88 参照) で、95のヴァリエーションがある。
入力方法は、以下の通りである。

1) タイルを1個ずつ配置する

Battlefield を作成するときの基本形で、[Edit] の [Plop mode on] を選択して行なう。但し、Battlefield Design Module の初期設定は、このモードがオンになっているので、わざわざ選ぶ必要はない。

(キーボードの場合) 配置するタイルを指定したら、☜をエディット・ウィンドに戻し、タイルを置く場所が、☜の指先に来るようにBattlefield をスクロールさせて、[↵]キーを押す。

(マウスの場合) タイルを指定し、配置する場所でクリックする。

2) 鉛筆書き機能を使う

これは、指定したタイルを鉛筆で線を描くように、連続して置いていく機能で、[Edit] の [Pen Down] を選択して行なう。
なお、このモードに入ると、エディット・ウィンドの中央に指定したタイルが反転して表示される。また、このモードを出るときは [Plop mode on] を選択する。
(キーボードの場合) タイルを指定してから、Battlefield をスクロールさせて置いていく。

※ マウスの場合は [Pen Down] を使わずに、タイルを指定してから、クリック・ボタンを押したまま、🖱を動かして書いたほうが機能的である。

3) 同一タイルで Battlefield 全体を埋め尽くす

タイルを指定してから、メニュー [Battlefield] の [Fill Map] を選択する。但し、この機能は Battlefield 全体を上書きしてしまうので、注意すること。

4) 同一タイルでエディット・ウィンド内を埋める

これは指定したタイルで、現在エディット・ウィンドに表示されている部分だけを埋める機能である。
タイルを指定してから、メニュー [Battlefield] の [Fill Screen] を選択する。

5) Battlefield に書き込んだものを全て削除する

メニュー [Battlefield] の [Clear Map] を選択する。

6) 直前の処理をキャンセルする

これは、最後に行なった処理をキャンセルして、その前の状態を復元する機能である。上に述べた５つの機能全てに対応しており、[Edit] の [Undo] を選択する。

★ ファイルのセーブ

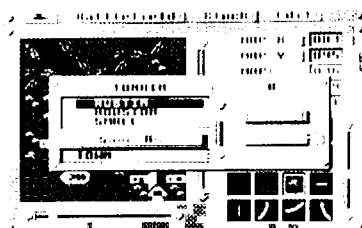
新しく作成した Battlefield をセーブするときは、[Battlefield]の [Save] を選択して下の画面に入り、ファイル名を入力してから

(キーボードの場合) [↑],[↓]キーで☞を [Save] に合わせて[↵]キー。

(マウスの場合) [Save] をクリック。

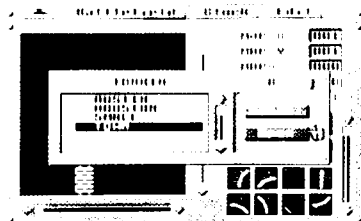
既存のファイルを修正して元のファイルにセーブするときは、

(キーボードの場合) [←],[→],[↑],[↓]キーで☞を元のファイル名に合わせてこれを反転させ、[↵]キーを押してから [Save] を選択する。

(マウスの場合) 元のファイル名をダブル・クリックしてから [Save] をクリックする。

★ ファイルのロード

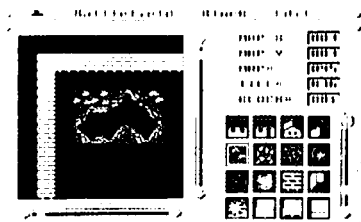
既存のファイルを呼び出すときは、[Battlefield] の [Load] を選択して下の画面に入り、

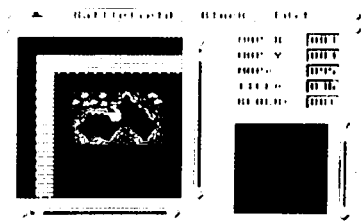
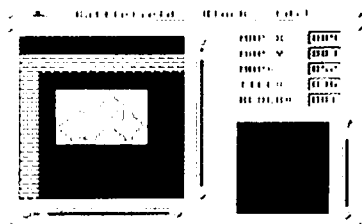
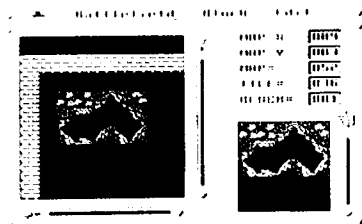
(キーボードの場合) [←],[→],[↑],[↓]キーで呼び出すファイル名を反転させてから、☞を [Open] に合わせて[↵]キーを押す。

(マウスの場合) 呼び出すファイル名をダブル・クリックする。

3．3．2 ブロックの使用方法

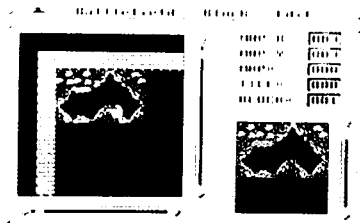
“ブロック”とは、複数のタイルが組み合わさってできたパーツのことであり、池や建物といったまとまりある地形を、Battlefield に書き込む場合、その効率を上げてくれる便利な機能である。
なお、ブロックの大きさはタイル数にして5×5を最大とする。
ブロック・ファイルの作成方法、及びその使い方は以下の通りである。

★ブロックの作成方法

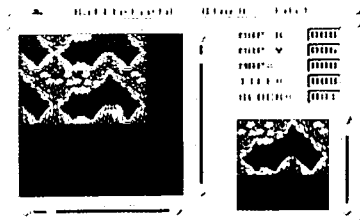
1) Design Battlefield を選択して新規のエディット・ウィンドを開き、ここでタイルを使ってブロックを作成する。

> ※ 既存の Battlefield の一部をブロック・ファイルにする場合は、その Battlefield をロードし、ブロックにする部分をエディット・ウィンド内にスクロールさせておく。

2) Edit の Display Blocks を選択する。この時点で、パーツ表示エリアが以下のように切り換わる。

3) Edit の Copy mode on を選択し、ブロック・ファイルにコピーする部分のタイルを反転させ、範囲指定を行なう。
(キーボードの場合) ☜をエディット・ウィンドウ内に合わせてから Battlefield をスクロールさせ、コピーするタイルを１枚ずつ☜の先に合わせて⏎キーを押す。指定解除は、再度⏎キーを押す。
(マウスの場合) コピーする範囲のタイルを１枚ずつクリックする。指定解除は再度クリック。
・指定を全て解除するときは、Block の Clear Copy を選択する。

4) (キーボードの場合) ☜をパーツ表示エリアに合わせ⏎キーを押す。
(マウスの場合) パーツ表示エリア内をクリックする。
・パーツ表示エリア内に指定したタイルが映し出され、コピー完了。表示エリア内のブロックの削除は、Block の Clear Block を選ぶ。
・なお、パーツ表示エリアをスクロールさせれば、最高４０個までのブロックを同一ファイル上にセーブすることができる。

5) ファイルのセーブ、及びロードは Block の Save と Load を選択して行なう。具体的操作は p.73 に準ずる。

★ブロックの入力方法

1) まず、ブロックを置く Battlefield をロードする。
Edit の Display Blocks を選択し、次に Block の Load で、書き込むブロックのファイルをロードする。この時点でエディット・ウィンドとパーツ表示エリア内の双方にブロックが映し出される。

> ※ ブロック作成後、即ち Copy mode on の使用後、引き続きブロックを Battlefield に書き込んでいく場合は、まず Plop mode on を選択してから行なう。

2) Battlefield をスクロールさせて、ブロックを置く位置に合わせ、
(キーボードの場合) をエディット・ウィンドに合わせてキー。
(マウスの場合) エディット・ウィンド内をクリック。
この時点で、ブロックが Battlefield 上に入力される。

3) 2)の操作を繰り返せば、他の場所にも入力することができる。

※ タイル入力モードに戻るときは、Edit の Display Tiles を選ぶ。

SECTION 4
“ＡＬＰＨＡ”の改良

SECTION BRIEF

戦闘シミュレーションを観てお判りの通り、“ＡＬＰＨＡ”の性能は決して十分なものとは言えない。最も大きな欠点は敵を追跡して射程内に入れながら、往々にしてその側を素通りしてしまうことである。原因はカプセル・ルーチン“Seek”の仕様にあるのだが、この節ではこれを他のカプセル・ルーチンと入れ替えることによって、より高度なＡＩを持つサイバータンク“ＢＥＴＡ”を作成する。
併せて、こうしたＡＩの修正や編集を極めて効率的に処理してくれるAI Module の持つ諸機能について解説する。

4.1 “ALPHA”の改良

1) Design を開いて Design Cybertank を選択し、“ALPHA”のAIをロードさせる。

> ※ OMEGA SYSTEM は Design Cybertank を選択すると、自動的に前回ロードしたサイバータンクのAIを表示するように設計されている。しかし、IDディスクにアクセスした直後など、この機能が働かなかったときは p.41 の操作方法にならって“ALPHA”のAIをロードする。

2) カーソルを“Do Seek”の“S”の前に持っていき、DEL キーで“Seek”を削除したら、替りに“Search”と打ち込む。
3) 同様に“Include Seek”も“Include Search”に修正し、正しく入力されているか、下とよく照らし合わせる。

```
Start
    Do Search
    Do Destroy
    Branch to Start

    Include Search
    Include Destroy
```

4) ここで Authorize する前に、メニュー Cybertank の Save as を選択して“ＢＥＴＡ”と打ち込み、新しいＡＩを“ＢＥＴＡ”の名前で Save する。この時点で“ＡＬＰＨＡ”とは別個のソース・ファイル“ＢＥＴＡ”が新しく開設される。

> ※ Save as の機能を利用してソース・ファイルを開設した場合、元のファイルは変更を受ける前の完全な状態で保存される。

5) Authorize の手続きを終えて ECM に戻ったら、シミュレーションの設計に入る。
メニューの Simulate を開いて Design a Simulation を選択し、各項目それぞれ以下のファイルを選択する。

Primary Tank	“ＢＥＴＡ”
Other Tanks	“ＡＬＰＨＡ”
	“ＶＩＰＥＲ”
Battlefield	“ＨＯＵＳＴＯＮ”

6) 次に、Save Simulation Design を選択し、“ＢＥＴＡＳＩＭ”の名前で Save する。
7) ECM に戻ったら、メニュー Simulate の Start a Simulation を選択して“ＢＥＴＡＳＩＭ”をロードし、“ＢＥＴＡ”の動きをよく観察する。

“ＢＥＴＡ”の動きをよく観察してみると、移動するごとに周囲を３６０°ていねいにスキャンしていることがわかる。この探知能力の向上によって、“ＢＥＴＡ”は敵を見逃してしまう確率をかなり低く抑えることが可能になったのである。

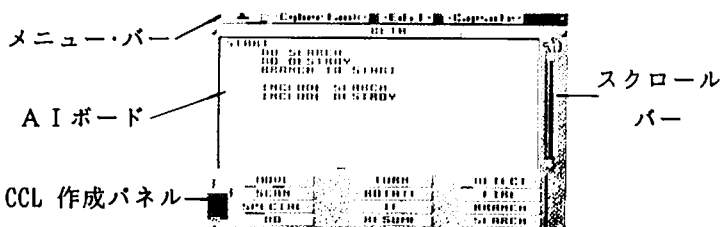
4.2 AI Module の諸機能

AI Module には、ＡＩの設計、或いは修正の作業効率化のために、さまざまな機能が備わっている。それをここでまとめて解説しておこう。特に４番目の“編集機能”は、諸君が今後何度も使用する大切な機能なので、確実に身につけておくこと。

では、これから実際に“ＢＥＴＡ”のＡＩを利用して、各機能を試していく。

Design を開いて Design Cybertank を選択し、“ＢＥＴＡ”のＡＩをロードする。

> ※これからの作業で“ＢＥＴＡ”のプログラムは壊されてしまうが、Design Control Module を出る際、“Save changes to BETA ?”に対して NO を選択すれば、元のＡＩを復元、保存することができる。

操作

（キーボード使用時）

★☞を、メニュー・バー，ＡＩボード，CCL 作成パネルの各欄に移動させるときは、ESC キーを押す。

★カーソルを移動させるときや、キーボードからコマンドを入力するときは、まず☞をＡＩボードに合わせてから行なう。

４．２．１　AI Module への自動アクセス

★ＡＩをより優れたものに仕上げていくためには、シミュレーションテストと、それに基づくＡＩの修正、これを繰り返していくより他に途はない。従って諸君も今後は同じＡＩを何回もロードして、それに修正を重ねていくのだが､OMEGA SYSTEM はこの際の手間を省くために、一旦 AI Module にアクセスした後は、Design Cybertank を選択しただけで自動的に AI Module を呼び出し、前回修正したＡＩをロードするように設計されている。

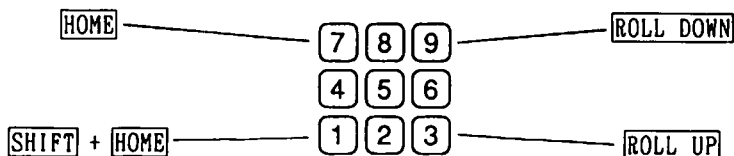
４．２．２　スクロール機能

★マウス使用者は、ＡＩボード右端にあるスクロール・バーの上下の小さな矢印をクリックすると、テキスト（プログラム・リスト）が１行ずつ上下にスクロールする。
また、スクロール・バーの四角い小さなレバーをクリックしたまま上下に引っ張れば、テキストの任意の場所に飛ぶことができる。

★キーボード使用者は ESC キーで☜をＡＩボード合わせ、↑,↓キーを押せば、上下にスクロールすることができる。

★以下の操作はマウス、キーボード共通。

ROLL UP キー ……………… １２行先に移動
ROLL DOWN キー …………… １２行前に移動
HOME キー ………………… テキストの先頭に移動
SHIFT + HOME キー ……… テキストの最後に移動

なお、テンキーは、カーソル・キーとして代用することもできる。

※ AI Module では、テンキーからの数字入力はできない。

4.2.3 インデント機能

★インデント機能とは既に述べた通り、⏎キーで改行をした際、カーソルが自動的に次の行の左端から数文字さがったところに移動する機能を指す。ＡＩコマンドは、ルーチンの名前である“ラベル”を除き、全てこの場所から入力しなければならない。

★１つのコマンドが画面の１行に納まらない場合、OMEGA SYSTEM は自動的に改行して、２行目以降をダブル・インデントした場所から続ける。これは単にプログラムの見やすさを配慮しただけではなく、⏎キーによる改行とは異なり、OMEGA SYSTEM がこの数行を１つのコマンドとして認識していることを表わすのである。では、実際に次のコマンドを“ＢＥＴＡ”のＡＩの後に入力してみよう。

If enemy tank was not found then branch to search

これは、フル・カスタムデザインでよく使用されるコマンドだが、“then”以降が自動的に改行してダブル・インデントすることが判る。では、同じコマンドを入力する際、仮に“found”の後で⏎キーを押して改行したらどうなるだろうか?
この場合、“then”以降をダブル・インデントした場所から入力したとしても、OMEGA SYSTEM は、“If enemy tank was not found”と“then branch to search” をそれぞれ別個のコマンドと認識してしまい、Authorize したとき双方に対してコマンドが不完全であるとのメッセージを出してくる。
即ち、OMEGA SYSTEM は ⏎キーによる改行をコマンドの終了と認識するので、１つのコマンドを入力している間は⏎キーによる改行を行なってはならない。

★カーソルを任意にインデントさせるときは [TAB] キーを、インデントをキャンセルするときは [BS] キーを押す。ラベル名を入力するときは [BS] キーを押して、カーソルを一番左端に戻す。

4.2.4　編集機能

テキストの削除、複写、或いは組み替えなどの編集機能を利用するときは、まず対象となるテキストの範囲指定を行なってから、メニューの Edit を開いて、各機能の項目を選択する。

★テキストの範囲指定

（キーボード使用者）は、対象となるテキストの頭にカーソルを移動させたら CTRL +S を1回押して最初の1文字を反転させ、後は ↑,↓,←,→ キーによって対象の全てを反転させる。指定をキャンセルするときは再度 CTRL +S を押す。

（マウス使用者）は、同じく対象部分の頭にカーソルを持っていき、そこでマウスボタンをクリックしたまま任意の方向に引っ張って、対象の全てが反転したところで指を離す。指定をキャンセルするときは、ＡＩボード上で、再度クリックする。

★複写

テキストのある部分を、複写して他の場所に書き写す場合、まずその部分を反転させてから Edit の Copy を選択する。この操作によって OMEGA SYSTEM はその内容をメモリー領域に一時的に保存する。次にカーソルを挿入箇所に持っていって Paste（貼込み）を選択する。Paste は、メモリー領域に書き込まれた内容をカーソルのある場所にロードするという機能を果たす。

更にカーソルを移動して Paste を選択すれば、同じテキストを何回もロードすることができる。

この機能を使って“ＢＥＴＡ”のプログラムの中の Do Search をコピーし、さっき入力した If enemy tank was not found ･･･ の後にロードしてみよう。

★削除

削除する部分を反転させてから、Cut を選択する。

この機能を使って“ＢＥＴＡ”のプログラムの中の次の２行

Include Search,　Include Destroy　を削除してみよう。

★移動

あるテキストの配置を変える場合は、そのテキストを Cut で削除してから、カーソルを移動先に持っていって Paste を選択する。
これは OMEGA SYSTEM が Copy 機能同様、Cut によって削除した内容も、一時メモリー領域に保存するシステムを活用したものである。
この方法で　Branch to Start　のコマンドを一番最後の行に移動させてみよう。

※メモリー領域に書き込まれたデータは、新たに Copy コマンドや Cut コマンドが選択されて、内容が書き改められない限り保存される。
※メモリー領域の記憶内容を書き換えることなくテキストのある部分を削除したい場合は、Edit の Clear を選択するか、もしくは DEL，BS キーを使用する。

★上書き

まず修正する部分を反転させ、そのままキーボードから新しいテキストを入力する。これを使えば古いテキストをいちいち削除しなくても新しいテキストに書き換えることができる。
この方法で、2行目の　Search　を　Seek　に書き換えてみよう。
また上書きする際、キーボード入力の代わりに Paste を選択すれば反転部分をメモリー領域の内容に書き換えることができる。

★復元

以上の編集機能は便利である反面、一度に広い領域を処理することができるので、操作を誤った場合は、せっかく組んだプログラムを失ったり、壊してしまうことになりかねない。この防御策として AI Module は、最後の処理をいつでもキャンセル ── 即ち処理を施す前の状態を復元するという "Undo" 機能を備えている。
この Undo は Edit メニュー最上部にあるが、表示が復元対象に応じてさまざまに変化するので、次にいくつか例を上げておこう。

誤った Paste をキャンセルする場合 ………………… Undo Paste
誤った Cut をキャンセルする場合 ………………… Undo Cut
誤ったキーボード入力をキャンセルする場合 …… Undo Typing

このように“Undo”機能は AI Module 上のあらゆる処理に対応しているので、今の処理をどうしてもキャンセルしたいと思った場合は、とりあえず Edit を開いて Undo を選択してみるとよい。

> ※“Undo”機能が働くのは、あくまでも直前の処理に対してのみである。従ってキャンセルしたい処理の後に何らかの処理をはさんでしまうと、“Undo”の対象は後の方の処理に移ってしまうので注意すること。

この機能を利用してテキストの一部を Cut で削除し、Undo Cut でもう一度復元させてみよう。
また、Undo ＊＊＊ を選択した後は、“Undo”項目の表示が“Redo”に変り、これを選択すると“Undo”の処理をキャンセルしてくれる。

以上述べた機能は、諸君がオリジナルのプログラムを組む際、極めて頻繁に使うものなので、完全にマスターするまで、実際に“ＢＥＴＡ”のＡＩ上でいろいろと試してみるとよい。

なお、これらの操作は以下のキーによって代行することもできる。

Copy（複写）………………… CTRL ＋C
Paste（貼込み）……………… CTRL ＋V
Cut（削除）…………………… CTRL ＋X
Undo（復元）………………… CTRL ＋Z

次はいよいよフル・カスタムデザインの研修に入る。

SECTION 5
フル・カスタムデザイン

SECTION BRIEF

セミ・カスタムデザインによるサイバータンクの設計を通じて、ＡＩの設計とはどんなものか、またサイバータンクとはどんな動きをするのか、だいたいのイメージは掴んでもらえたと思う。この節ではいよいよ CCL を使った"フル・カスタムデザイン"の基本的なノウハウの研修に入る。

フル・カスタムデザインでＡＩを組むために、把握しなければならないことは、要約すると

1) サイバータンクはいかなる機能で構成されているのか
2) それを機能させるための CCL コマンドのヴァリエーションにはどのようなものがあるか
3) これをどのように配列すれば設計者の意図通りにサイバータンクを動かすことができるのか

の３点である。今後の研修は以上の項目に沿って進めるが、2)の CCL コマンドのヴァリエーションについては、必要最小限のコマンドでとりあえずＡＩをひとつ組み上げ、諸君が CCLコマンドに慣れた段階で解説することにする。

5．1　Battlefield の空間概念

諸君は今後 Combat Simulation Module や Cybertank Test Module を活用してＡＩの弱点を解明し、その改善を図っていくのだが、戦闘シミュレーションが行なわれる Battlefield の空間概念に対する理解がなければ、フル・カスタムデザインで使用される CCL コマンドの意味を理解することはできない。以下、これについて解説する。

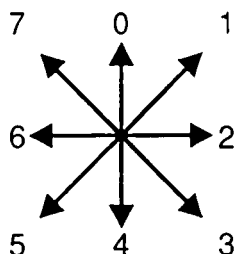
5．1．1　座標系

★Battlefield（外枠の壁を含む）は、縦横それぞれ ６４×６４ のマス目で区切られており、この１マスを“セル”と呼ぶ。

★セルは、１辺が 1 hm（=100m）の正方形を成している。

★OMEGA SYSTEM はセルを単位とする座標系を持ち、各セルの位置は、Battlefield 左上の角を原点とする ｘ座標(0～63)，ｙ座標(0～63)の値で認識される。但し、外枠の壁は幅：１セル、かつ固定なので Battlefield 内側の座標は ｘ，ｙ 共に(1～62)で表わされる。

5．1．2　方位

★OMEGA SYSTEM が Battlefield のデータ管理をセル単位で行なっているため、方位の概念もこれに拘束される。即ち、Battlefield における方位は下図の通り、１つのセルを取り囲む８つのセルの方角に限られ、この８方位を“絶対方位”と呼ぶ。

★４５°を単位とするこの“絶対方位”は、北を“０”として時計廻りにそれぞれ“０～７”のコードナンバーを持ち、OMEGA SYSTEM は方位の認識をこのコードナンバーによって行なう。

５．１．３　距離

★隣接するセル間の距離は、垂直、水平、斜め、いずれの方向も１hm とする。従って、Battlefield 内側の１辺の距離と、対角線の距離は双方共に 62hm であり、この移動に要する時間も等しいものとする。スキャナーの探査領域や、砲弾の射程領域を示すスクリーンが円形でなく、正方形で表わされるのはこのためである。

５．１．４ 構成要素

Battlefield を構成する各セルの情報は、以下のような定義に従ってグループ分けされ、OMEGA SYSTEM は、その“タイプ・ナンバー”によってこれを管理する。

（タイプ・ナンバー）	（定義）
1	道路，草地，石，etc
2	水域
3	破壊されたタンク，潰された木
4	立木
5	家屋，指令部(ＨＱ)
6	ビルディング，境界壁
7	サイバータンク

次項で解説する“移動センサー”“スキャナー”などの探査装置は、対象の認識を、全てこのタイプ・ナンバーによって行なっている。

5．2　サイバータンクの基本機能

サイバータンクの持つ基本機能は、大きく分けると

移動
探査
攻撃

の３つに分類することができ、このうち探査機能は、“移動センサー”と“スキャナー”という、２つの異なる装置によって支えられている。では、個々について詳しく見ていこう。

5．2．1 移動

★車両の移動は、ｘ座標，ｙ座標 の変化として、これを認識する。
★車両が直接移動できる方角は絶対方位の８方向に限られる。
★１hm の移動に要する時間は、車体の重量、搭載する Drive System のグレードによって異なる。
★車両の移動は、方向転換を行なう“Turn”コマンドと、前後への移動を行なう“Move”コマンドによって、これを実行する。

5．2．2 移動センサー

★移動センサーは、車両周辺の“obstruction”(障害物)を探査する装置で、距離：３hm，幅：１hm の範囲を探査する。
★探査域内に複数の障害物が存在する場合、移動センサーは最も近くに位置する障害物を探知する。
★移動センサーは絶対方位の８方向全てを探査することができる。
★移動センサーは、これを使用する際、必ず探査すべき方角を指定してやらなければならない。
★移動センサーは“Detect”コマンドによって機能する。

５．２．３ スキャナー

★スキャナーは、以下３つの目標物を探査することができる。

1) Enemy tank　　（敵車両）
2) Enemy HQ　　　（敵指令部）
3) Closest object（最も近い物体）

★スキャナーの探査域は、コンポーネントのグレードによって異なり、距離：２０～５０hm，掃引角：３０～９０°の範囲を探査する。
★スキャナーは絶対方位の８方向全てを探査することができる。
★スキャナーは、これを使用する際、必ず探査すべき目標物を指定してやらなければならない。
★スキャナーは、“Scan”コマンドによって機能する。
★スキャナーは、最後に探査した方角を絶対方位のコードナンバーによって記憶しており、新らたにスキャナーを旋回させる命令を受けるまで、車両の旋回や移動に影響されることなく、その方角を維持し続ける。
★スキャナーの旋回は“Rotate”コマンドによって、これを実行する。
★スキャナーは、目標物が障害物の背後に入ると、これを探知することができない。(p.95 参照)

５．２．４ 攻撃

★攻撃は、いかなる対象に対しても、これを実行することができる。
★全ての兵器は、射程：４hm である。
★砲弾の供給は無制限であるが、攻撃によって一定の燃料が消費される。
★弾頭の総合的な破壊力や、各部位への損傷効果は兵器の種類によって異なる。
★攻撃は、絶対方位に拘束されることなく、射程内であればいかなる方角に対しても、これを実行することができる。
★攻撃は“Fire”コマンドによって、これを実行する。

5.3　ＡＩの基本構造の決定

サイバータンクの持つ機能を一通り理解したところで、ＡＩの基本構造の立案に入ろう。
目的は敵車両の破壊。サイバータンクの行動を決定する判断材料は、移動センサーとスキャナーから得られる情報のみである。

★ 戦闘開始

1) 探査

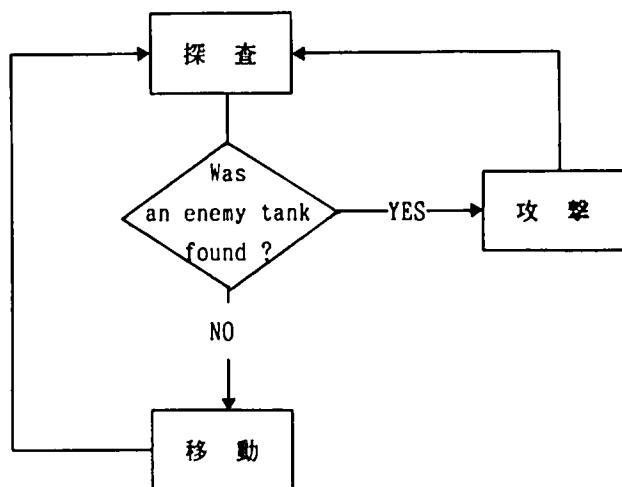
先制攻撃を受けてはたまらない。まずスキャナーで敵が近くに迫っていないか確認しよう。
前方を探査。敵がいなければスキャナーを左に１旋回させて再び探査。これを繰り返し、途中で敵を発見すればその時点で 3) の攻撃に分岐し、８方位(360°)全て探査し終わった時点でなおも敵を発見することができなければ 2)に分岐して１hm 前進し、そこで再び探査を開始する。
こうして敵を発見するまで 1), 2)の行動を繰り返す。

2) 移動

移動する際には、移動センサーでまず進行方向に障害物が存在しないか確認をする。障害物に衝突すれば、装甲などに少なからぬダメージを受けるからである。
障害物がなければそのまま前進。あれば砲撃によって障害物を除去して前進。除去できないときは方向転換して、移動可能な方角に前進する。移動が完了したら 1) に分岐。

3) 攻撃

遂に敵を発見。再度スキャナーで敵の位置を確め、万一敵が逃走していたら 1)に戻る。射程内であれば間髪容れずに攻撃。射程外であれば接近して射程に入れ次第攻撃開始。敵を破壊するか、或いは敵が逃走してスキャナー・サイト上から消えたら 1)に還る。

以上は諸君がこれから組むＡＩの全容を文章で表わしたものであり、
下の(図１)はこれを簡単なフローチャートに描いたものである。
フローチャートにしてみると、このＡＩが完結した“循環構造”を持っていることがよくわかる。

（図１）

このＡＩは、“探査”“移動”“攻撃”という３つのルーチンから構成される非常にシンプルなものであるが、あらゆるＡＩの基本となる大変優れた構造を持っている。
では各ルーチンごとに、詳しいフローチャートを作成し、それに基づいてＡＩを組んでいこう。

ECM に戻ったら、メニュー Design の Design Cybertank を選択し、Design Control Module に入る。次に Cybertank の New を選択し、“ＧＡＭＭＡ”の名前で新規にサイバータンクを開設する。
コンポーネントは予算内で各自好きなものを選び、最後に AI を選択して、AI Module に入る。

（図１）の実行順序にならえば、探査ルーチンから着手するところだが、最初の探査活動でスキャナーの域内に敵が存在しなかったことを想定し、まず移動ルーチンから始めよう。

5．4　移動ルーチン

移動ルーチンを組むときの最大の課題は、障害物に対していかに対処するかということである。このことは逆から言えば、障害物の概念規定が移動ルーチンを大きく左右しているということに他ならない。
移動ルーチンのフローチャート作成に入る前に、ここで移動センサーやスキャナーが Battlefield 上の物体をどのように認識するか､またそれらの物体の特性は何かについて整理しておこう。

5．4．1 Battlefield の構成要素の特性

以下は、p.89 に示したのと同じ Battlefield の構成要素である。

（タイプ・ナンバー）	（定義）
1	道路，草地，石，etc
2	水域
3	破壊されたタンク，潰された木
4	立木
5	家屋，指令部（ＨＱ）
6	ビルディング，境界壁
7	サイバータンク

★ 移動センサーは、タイプ・ナンバー２以上を全て“obstruction”（障害物）として認識する。

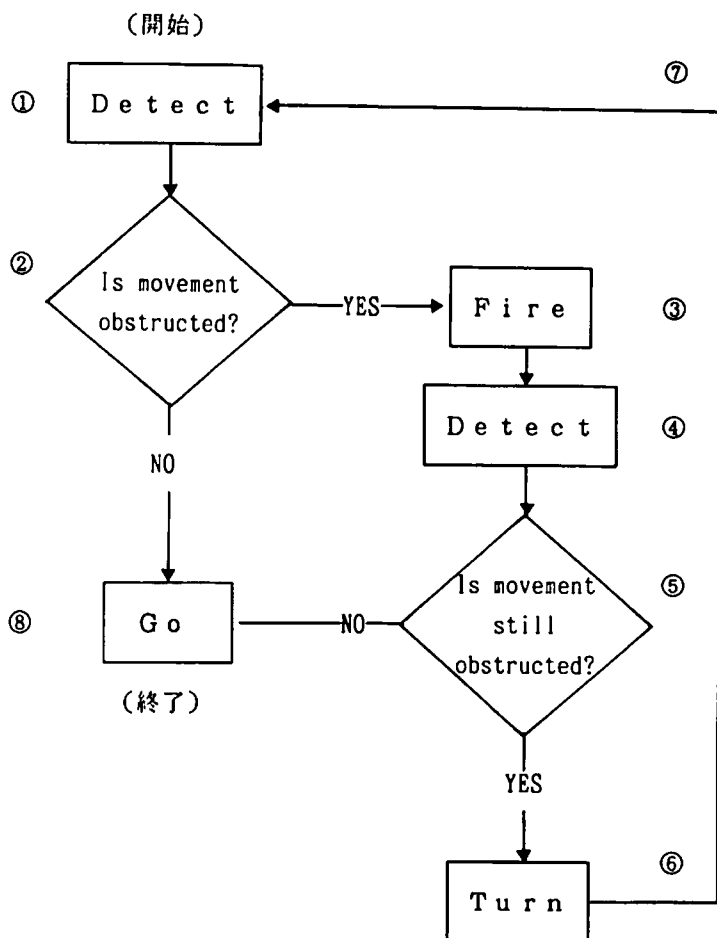
★ スキャナーは、３以上を全て“closest object”として認識する。

★ スキャナーの目標物が４以上の背後に入ると、スキャナーはこれを探知することができない。

★ ３～５は１回の砲撃によって破壊することができるが、２と６は砲撃によって破壊することはできない。

★ 障害物に衝突するとタンクは一定のダメージを受ける。但し、水陸両用車は水域に入ってもダメージを受けることはない。

5.4.2 移動ルーチンのフローチャート作成

以下はP.92の 2)の文章をフローチャートに描いたものである。

(図2)

解説

① 移動センサーで、進行方向（tank direction）に障害物がないか探査する。探査域内に障害物を発見すると OMEGA SYSTEM は自動的にその位置とタイプ・ナンバーをメモリー領域に記憶する。

② 進行方向への移動が①で探知した障害物によって妨げられているかを判断する。障害物がなければ⑧に分岐。障害物があっても、次の移動に支障がなければ（即ち、障害物が直前に迫っていなければ）、同じく⑧に分岐。直前に迫っている場合のみ③に分岐する。

③ 障害物に対して砲撃を１回くわえる。

④ ①を繰り返して、砲撃の結果を確認する。即ち、障害物は１回の砲撃によって破壊できるものと、砲撃による破壊が不可能なものとの２つに分れるので(p.95 参照)、①で探知した障害物が破壊されずに残っていれば、それは破壊不能な障害物とみなすことができる。

⑤ ②の判断を繰り返す。移動を妨げる障害物があればそれは破壊不能な障害物なので、⑥に分岐して回避。移動を妨げる障害物がなければ、それは障害物が破壊されたことを意味するので⑧に分岐。

⑥ 車両を左に１旋回（右でも可）。

⑦ ①に分岐して再びこの行程を繰り返し前進可能な方向を探る。

⑧ １hm 前進。

この移動ルーチンは①の“Detect”に始まって、⑧の“Go”で１行程を完了するものである。従ってサイバータンクは移動可能な方角を見つけ出して１hm 前進するまで、このルーチンを出ることはできない。次は①〜⑧のナンバリングに沿って、これを CCL に翻訳していく。

５．４．３ 移動ルーチンのＡＩ作成

①～⑧を CCL に翻訳したのが下のリストである。これをキーボードから入力する。

```
        Smartmove
①==>        Detect obstruction at tank direction
②==>        If movement is not obstructed then branch to Go
③==>        Fire weapon at obstruction
④==>        Detect obstruction at tank direction
⑤==>        If movement is not obstructed then branch to Go
⑥==>        Turn tank left 1
⑦==>        Branch to Smartmove

        Go
⑧==>        Move tank forward 1
```

解説

★ “Smartmove” はこのルーチン全体を指すラベル名であり、ラベル “Go” はこれに含まれる。

> ※コマンドの処理(実行)は、特別の指示がない限り１行目から順番に行なわれる。
> 例えば②の条件節が満たされない場合は、どこそこに分岐しろという指示がなくても、OMEGA SYSTEM は自動的に次の行の③の処理に移るよう設計されている。

５．４．４ 移動ルーチンのシミュレーション

移動ルーチンを組み終えたところで、このＡＩが意図した通りに設計されているか Combat Simulation Module を使って確認してみよう。以下の指示に従って作業を進める。

1) このままでは、移動ルーチンはその行程を１回しか実行してくれない。 Move tank forward 1 の後に “Branch to Smartmove” というコマンドを入力して、このルーチンに循環構造をもたせる。
2) Authorize を選択して認定に成功したら、データをセーブする。
3) Simulate の Design a Simulation を選択し、次の仕様でシミュレーション·デザイン·ファイルを作成し、“ＧＡＭＭＡＳＩＭ”の名前でセーブする。

Primary Tank ………… “ＧＡＭＭＡ”
Other Tanks ………… “ＡＬＰＨＡ”
“ＢＥＴＡ”
Battlefield ………… “ＨＯＵＳＴＯＮ”

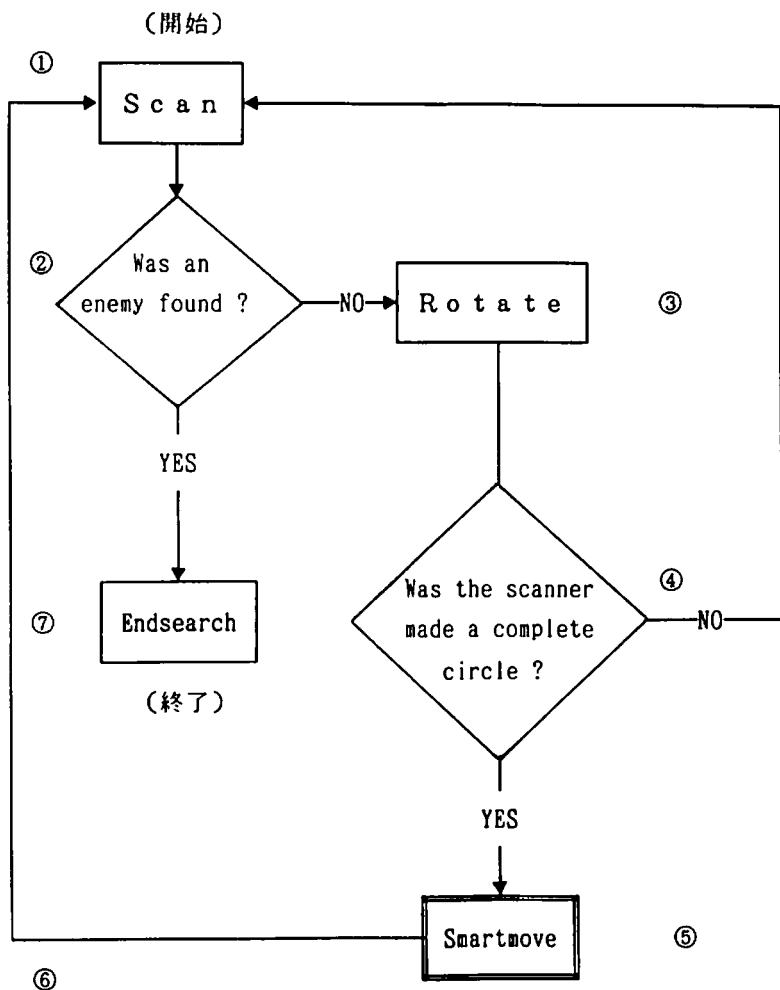
4) Simulate の Start a Simulation を選択し、“ＧＡＭＭＡＳＩＭ”をロードし、その動きを観察する。

シミュレーション開始後、タンクが障害物手前まで直進し、障害物を砲撃。障害物が破壊されればそのまま直進、破壊できなければ左に順次旋回して移動可能な方向に前進。これを繰り返すようであれば、意図通り設計されていると判断することができる。

５．５ 探査ルーチン

探査ルーチンを作成するうえで最も肝要なのは、常に周囲３６０°を漏れなく探査するということである。これを怠ると敵の発見が遅れ、先制パンチをくらうはめになる。

5.5.1 探査ルーチンのフローチャート作成

以下は p.92 の 1)の文章をフローチャートに描いたものである。

(図3)

解説

① スキャナーで進行方向の敵を探査する(シミュレーション開始時、スキャナーは常にタンクの進行方向を向いている)。敵を探知すると OMEGA SYSTEM は探知した時点での敵の位置を記憶する。

② 敵を探知した場合は、⑦に分岐してこのルーチンを終了し、探知できなかった場合は③に分岐する。

③ 別の方角を探査するために、スキャナーを左に1旋回させる(右でも可)。

④ ここでは、スキャナーがこの場所での探査を8方位全て完了したかどうかを判断する。具体的な判断方法は、この時点でスキャナーがタンクの進行方向に向いているか否かをみる。同じ方向を向いていればスキャナーが一周して元の向きに帰ってきたことを意味するからである。360°探査し終わっていれば⑤に分岐して移動し、終わっていなければ①に戻って新しい方角を探査する。

⑤ 移動ルーチン、即ち“Smartmove”を実行する。

⑥ “Smartmove”で移動終了後、再び“Ｓｃａｎ”に分岐する。

⑦ このルーチンを終了する。

このルーチンは①の“Ｓｃａｎ”に始まって、⑦の“Endsearch”で1行程を終了するものである。従ってサイバータンクは敵を発見するまでこのルーチンを抜け出すことはできない。
また、前節で作成した“移動ルーチン”が、実は“探査ルーチン”の1行程に組み込まれていることが、このフローチャートから読み取れる。

5.5.2 探査ルーチンのＡＩ作成

①〜⑦を CCL に翻訳したのが、以下のリストである。キーボードで移動ルーチンの後に続けて入力する。

```
        Search
①==>        Scan for enemy tank
②==>        If enemy tank was found then branch to Endsearch
③==>        Rotate scanner left 1
④==>        If tank is not aligned with scanner then branch to
                Search
⑤==>        Do Smartmove
⑥==>        Branch to Search

        Endsearch
⑦==>        Resume
```

解説

★ “Search” はこのルーチン全体を指すラベル名で、“Endsearch” はこれに含まれる。

★ “ Do Smartmove ” は、セミ・カスタムデザインでも見たように、“ Smartmove というラベル名のルーチンを実行せよ” という意味のコマンドである。
この命令によって OMEGA SYSTEM はこのＡＩ上に “Smartmove” のラベル名を捜し出し、そのルーチンを実行する。

★ OMEGA SYSTEM は “Smartmove” を実行終了後⑥に還って来て、その指示通り “Search” に分岐する。

★⑦の “Resume” に関しては後程解説する。

5.5.3 探査ルーチンのシミュレーション

では次に、探査ルーチンを繰り返し実行させるためのＡＩ（仮にこれを“継続探査ルーチン”と呼ぶ）を組んでみよう。

1) 一番初めの行にカーソルを戻して、ラベル“Smartmove”の前に

```
Start
    Do Search
    Branch to Start
```

という３行を挿入する。
ラベル名“Start”のこのルーチンは“Search”終了後、“Start”に分岐して再び“Search”を実行する。

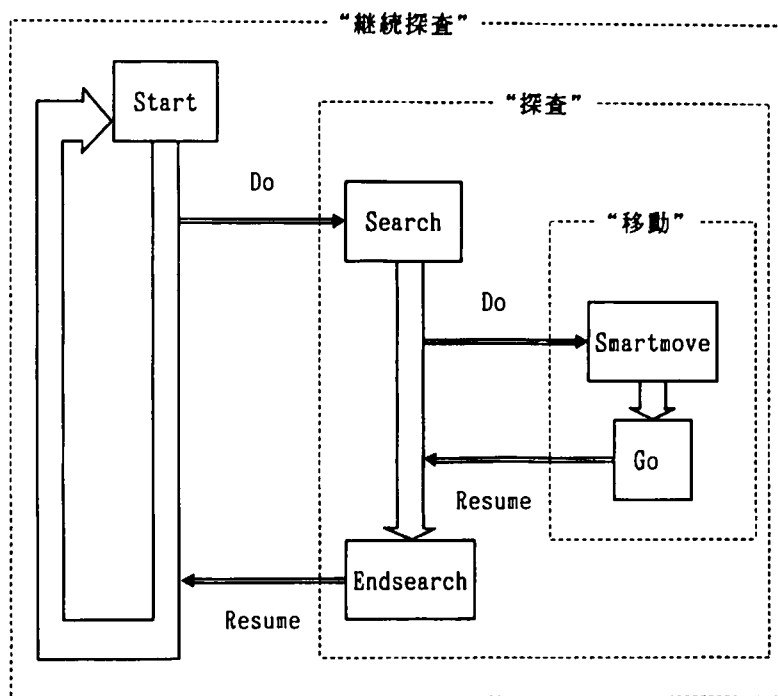
2) p.99 で、移動ルーチンの最終行に入力した Branch to Smartmove を削除し、替りにその行に Resume（後述）と入力する。
3) 正しく入力されているか p.107 とよく照らし合せる。
4) Authorize を選択して認定に成功したら、“ＧＡＭＭＡ”のＡＩをセーブする。
5) Simulate の Start a Simulation を選択し、“ＧＡＭＭＡＳＩＭ”をロードし、その動きを観察する。

ここで、これまでの作業を一度整理しておこう。

5.5.4 継続探査ルーチンの構造

諸君はこれまでに３つのルーチンを組んできた。

1) 移動ルーチン ………… ラベル“Smartmove”で始まり、移動可能な方向に１hm 前進することによってルーチンを終了する。
2) 探査ルーチン ………… ラベル“Search”で始まり、敵を発見することによってルーチンを終了する。
移動ルーチンをその行程に含む。
3) 継続探査ルーチン …… ラベル“Start”で始まり、探査ルーチンを繰り返し実行する。

この３つのルーチンの関係は、図で表わすと次のようになる。

（図４）

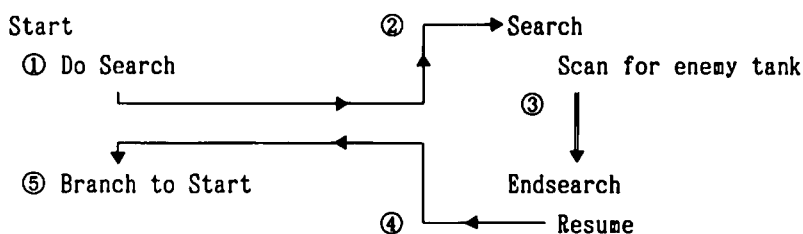
この図で注目してもらいたいのは、継続探査ルーチンが循環構造を持つのに対して、探査、移動の両ルーチンには"始め"と"終わり"があるということである。
では、"Do Search"，"Do Smartmove"というコマンドによって処理を開始するこの２つのルーチンはその一行程を終了した後、いかなるコマンドによって、どの行の処理に移行するのであろうか? 実はこの点に関する正確な説明はまだ行なわれていない。次はこれを処理してくれるコマンド"Resume"について解説する。

5.5.5 "Do" と "Resume"

"Do" と "Resume" に注目して（図4）をもう一度見てみよう。

"Start" から "Search" へのアクセスは	"Do"
"Search" を抜けて "Start" に戻る時は	"Resume"
"Search" から "Smartmove" へのアクセスは	"Do"
"Smartmove" を抜けて "Search" に戻る時は	"Resume"

このように "Do" と "Resume" は必ず対になって機能するもので、概念的にはルーチン間の接続を司どるコマンドと定義することができる。では "Resume" とは具体的にどのように機能するのであろうか？

"Resume" という言葉の原義は "一度中断したものを再開する" というもので、コマンド "Resume" は "Do" コマンドによってアクセスしたルーチンを終了させ、元のルーチンを再開させるという機能を持つ。これを "Start" と "Search" の間の処理手順で追って見ると——

解説

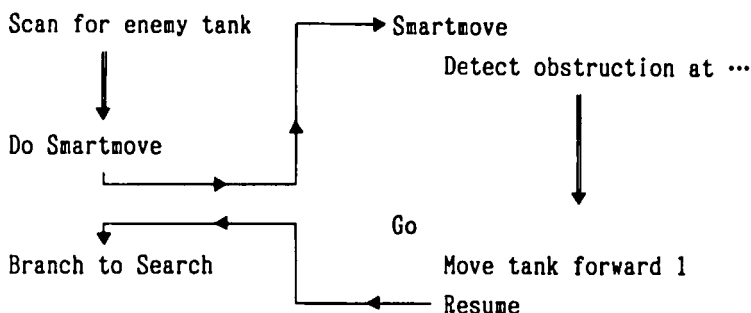
①：OMEGA SYSTEM はまず "Start" の１行目のコマンドを処理する。

②：コマンド処理は本来上から順番に行なわれるので、次は２行目の⑤が処理されるところだが、①の指示があるのでこちらの処理を一時中断して "Search" に飛ぶ。

③：ここで探査ルーチンを処理し、

④："Resume" に逢ったところで探査ルーチンを抜けて、"Start" 以下のコマンド列（元のルーチン）に復帰し、

⑤：本来①の次に処理されるはずだったこの行から処理を再開する。

同様に、探査ルーチンと移動ルーチンの関係では——

という流れにになる。

このように“Do”コマンドを使用するときは、OMEGA SYSTEM が、このコマンドによって呼び出されたルーチンを処理した後、同ルーチン内の“Resume”によって“Do ＊＊＊”の次の行に還ってくることを前提として使用しなければならない。
逆を言えば“Do”コマンドによって呼び出されることを想定しているルーチンは、終了と同時に元のルーチンに戻れるよう、必ず“Resume”コマンドを内に含んでいなければならない、ということになる。

実例として“ＡＬＰＨＡ”のＡＩに使用した“Seek”“Destroy”に Resume が含まれているか、p.51, 52 を見て確かめてみよう。

“Seek”には Resume が２回出てくるが、ラベル“Go”の Resume はコマンド“Do Onward”を受けるもので、“Seek”ルーチンを抜ける機能を果たしているのはラベル“Look”の Resume である。

ここまでの作業で、“ＧＡＭＭＡ”のＡＩは、敵を発見するまで戦場を捜し回るという能力を得た。残るは攻撃能力だけである。

“継続探査ルーチン”

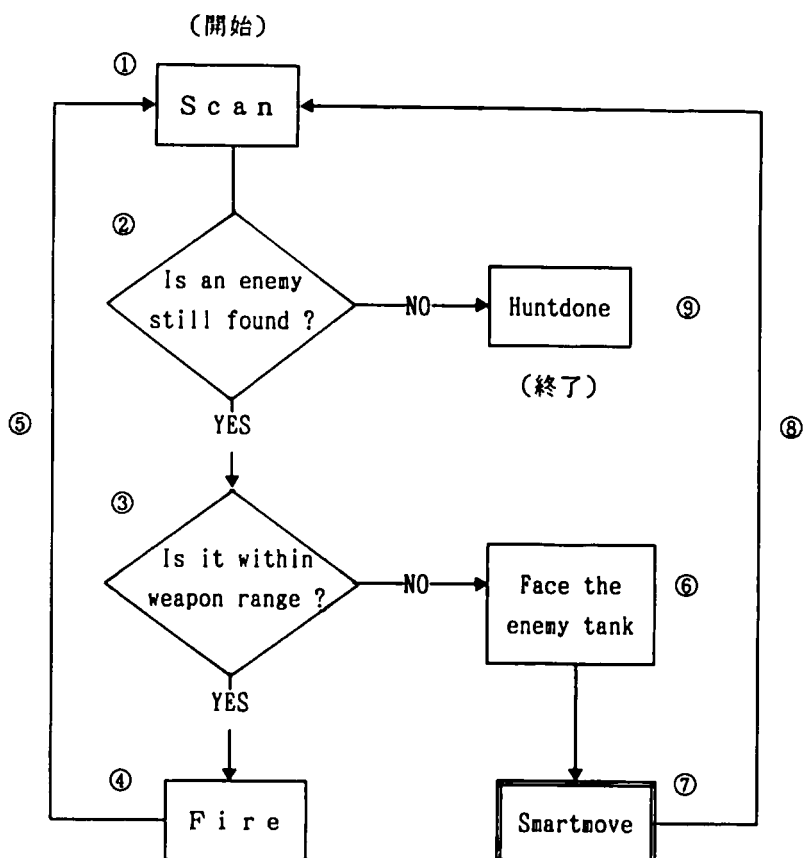
```
Start
    Do Search
    Branch to Start

Smartmove
    Detect obstruction at tank direction
    If movement is not obstructed then branch to Go
    Fire weapon at obstruction
    Detect obstruction at tank direction
    If movement is not obstructed then branch to Go
    Turn tank left 1
    Branch to Smartmove
Go
    Move tank forward 1
    Resume

Search
    Scan for enemy tank
    If enemy tank was found then branch to Endsearch
    Rotate scanner left 1
    If tank is not aligned with scanner then branch to
        Search
    Do Smartmove
    Branch to Search
Endsearch
    Resume
```

5.6　攻撃ルーチン

5.6.1 攻撃ルーチンのフローチャート作成

以下は p.92 の 3)をフローチャートにまとめたものである。

(図5)

解説

① スキャナーで再度探査して、敵が逃走していないかを確認する。

② 敵の存在が確認できたら次は③に分岐して、射程内か否かをみる。また、もしも敵の機影がスキャナー・サイト上から消えてしまっていた場合、即ち敵が探査域の外に逃走してしまったか、もしくは完全に破壊されてしまった場合は、⑨に分岐してこのルーチンを終了する。

③ 敵が射程内であれば④に分岐して攻撃。射程外であれば⑥に分岐して追跡の体制をとる。

④ 敵車両を砲撃。

⑤ ①に戻ってこの行程を繰り返し、敵を見失うか、或いは敵を完全に破壊するまで攻撃を続ける。

⑥ 敵に接近するため、敵のいる方向に車両の向きを変える。

⑦ “Smartmove” に飛んで移動ルーチンを実行。

⑧ ⑦の前進で敵が射程内に入ったかどうかを調べるため、直ちに①に分岐する。

⑨ 攻撃ルーチンを終了し、元のルーチンに還る。

５．６．２ 攻撃ルーチンのＡＩ作成

キーボードで以下のリストを今まで組んできたＡＩの後に続けて入力する。

```
          Hunt
①==>          Scan for enemy tank
②==>          If enemy tank was not found then branch to Huntdone
③==>          If enemy tank is beyond weapon range then
                   branch to Closein
④==>          Fire weapon at enemy tank
⑤==>          Branch to Hunt

          Closein
⑥==>          Turn tank to face enemy tank
⑦==>          Do Smartmove
⑧==>          Branch to Hunt

          Huntdone
⑨==>          Resume
```

解説

★ "Hunt" はこのルーチン全体を指すラベル名で、 "Closein" も "Huntdone" もこれに含まれる。

次は、この攻撃ルーチンを継続探査ルーチンに組み込む設計の最終段階に入る。

5.6.3 “GAMMA”のＡＩの作成

“ＧＡＭＭＡ”のＡＩが敵の発見、及び破壊というサイバータンクの基本任務を果たすためには、攻撃ルーチンをどのようにして継続探査ルーチンに組み込んでやればよいであろうか?

★その具体的方法を知る第一の手掛りは攻撃ルーチンの中にある。
　攻撃ルーチンの構造を振り返ってみると、それは敵を破壊するか、もしくは見失ってしまった時点で、元のルーチンに“Resume”する構造になっている。従って攻撃ルーチンは“Do”コマンドによって呼び出されることを前提としていることが判る。攻撃ルーチンを表わすラベル名は“Hunt”なので、それを実行させるためのコマンドは“Do Hunt”でなければならない。
★ではこの“Do Hunt”をどこに挿入するか?
　もう一度、ＡＩの基本構造に立ち返ってみよう。p.93 の（図１）にあるように、攻撃ルーチンへのアクセスは敵の発見を前提としている。一方、探査ルーチンは敵を発見した時点で“Resume”する構造になっている。従って探査ルーチンの“Resume”を受けて、直ちに“Do Hunt”が処理されるようなＡＩを組めばよい。
★こうして攻撃ルーチンが実行され、敵を破壊するか、或いは見失って“Resume”した後は、再び探査ルーチンに還って敵の探索に向う。

従ってラベル“Start”のコマンド列は以下のようになる。

```
Start
    Do Search
    Do Hunt
    Branch to Start
```

これで遂に、フル・カスタムデザインによる“ＧＡＭＭＡ”の設計は完了である。 Authorize して設計ミスがないかチェックしてみよう。

早速、戦闘シミュレーションにかけたいところだが、最後にプログラミング・テクニックを上達させるうえで、非常に大切な概念をここで押さえておこう。

５．６．４　“ＧＡＭＭＡ”のＡＩの構造

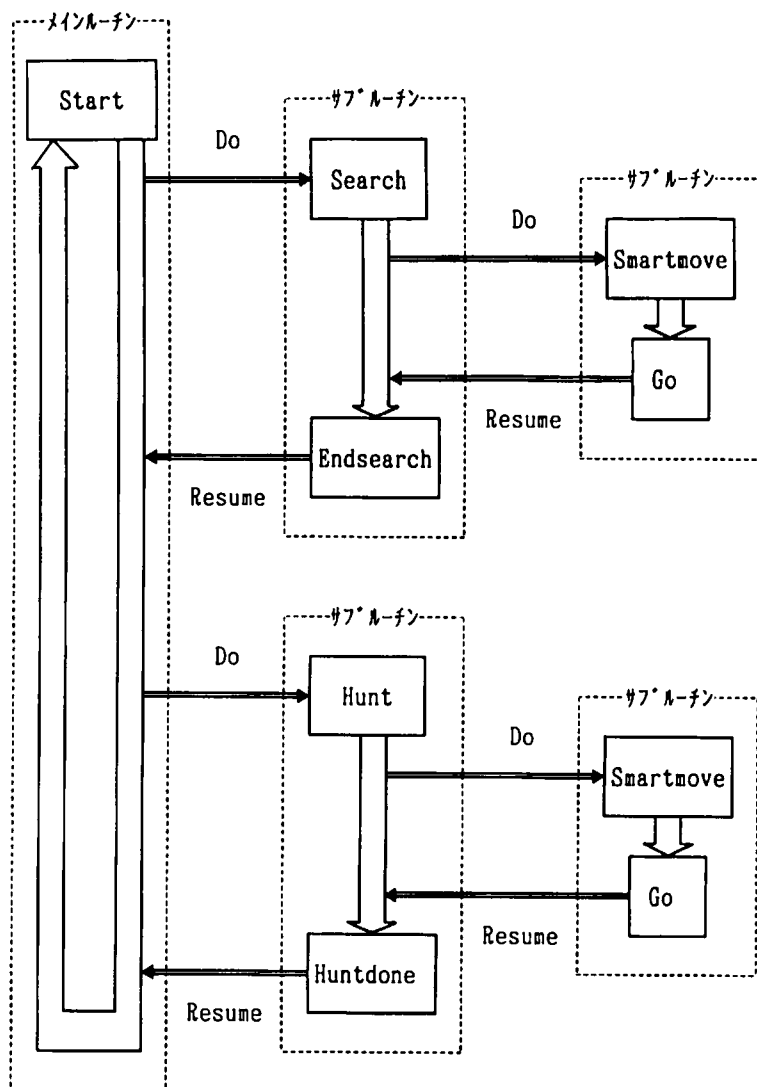
右の（図６）は“ＧＡＭＭＡ”のＡＩの構造を図示したものである。
“ﾒｲﾝﾙｰﾁﾝ”として破線で括られた部分に注目してみよう。この部分はラベル“Start”に続く３つのコマンド、即ち

```
Do Search
Do Hunt
Branch to Start
```

を指しているのだが、この３行が果たしている実質的な役割は何であろうか? いま仮に“ＧＡＭＭＡ”のＡＩを次のように書き換えたとして考えてみよう。（右の図を参考にしてＡＩの流れをイメージする）

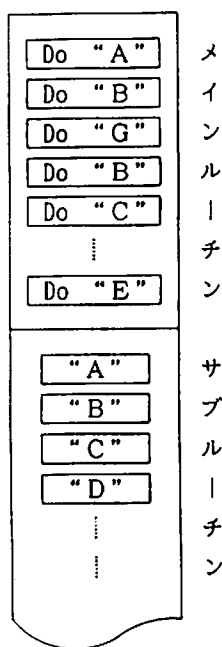
★まずラベル“Start”以下の４行を削除して、ＡＩをラベル“Search”から、即ち探査ルーチンから直接書き始める。
★“Endsearch”に来たら、“Resume”の代りに“Branch to Hunt”と入力して直接攻撃ルーチンにつなげてやる。
★攻撃ルーチンを経て“Huntdone”に来たら、“Resume”の代りに“Branch to Search”として直接最初の探査ルーチンに還してやる。

こうすれば、“ﾒｲﾝﾙｰﾁﾝ”として括られた部分がなくても、このＡＩは“ＧＡＭＭＡ”のそれと全く同じ機能を果たすばかりでなく、プログラムも４行短縮することができるのである。
では、なぜわざわざラベル“Start”を設けたのか?
この疑問は即ち、“Do, Resume”コマンドと“Branch to”コマンドが果たすべき機能の本質的な違いは何かということに他ならない。
これに明解な説明を与えてくれるのが、メイン・ルーチンとサブ・ルーチンの概念である。

ﾒｲﾝﾙｰﾁﾝ
Start
Do
ｻﾌﾞﾙｰﾁﾝ
Search
Do
ｻﾌﾞﾙｰﾁﾝ
Smartmove
Go
Resume
Endsearch
Resume
ｻﾌﾞﾙｰﾁﾝ
Do
Hunt
Do
ｻﾌﾞﾙｰﾁﾝ
Smartmove
Go
Resume
Huntdone
Resume

（図６）

5.6.5 “メイン・ルーチン”と“サブ・ルーチン”

結論から先に言えば
“サブ・ルーチン”とは“Do”コマンドによって実行され、“Resume”によって終わる、或る特定の任務を果たすひとつのルーチンで、
“メイン・ルーチン”とはこれらサブ・ルーチンの処理順序を指示することによって、実際にＡＩの構造を決定しているところのルーチンと定義することができる。

この関係をわかり易く図解したのが下の（図7）である。
このように、プログラムをメイン・ルーチンとサブ・ルーチンの概念に則って構成することをプログラムの“構造化”と言うが、この構造化には優れた特徴がいくつかあるので、それについて解説しよう。

（図7）

★プログラミングの効率化
プログラムを組んでいると、よく異なる場面で同じルーチンを必要とすることがある。例えば“ＧＡＭＭＡ”の中の移動ルーチンがこれに当るが、このように使用頻度の高いルーチンは一か所に置いて必要なときに“Do”コマンドで呼び出すようにしてやれば、何行何十行にも及ぶリストをその都度書く手間が省ける（たとえ１度きりしか使用しないルーチンでも、メイン・ルーチンには書き込まず、サブ・ルーチンとして外に置くほうが望ましい）。
p.112 の書き換えによってプログラムを短縮することができたのは特殊な例であり、ＡＩが複雑になればなるほど構造化によるプログラムの圧縮はその効果を上げるのである。

★全体像の把握しやすいプログラム

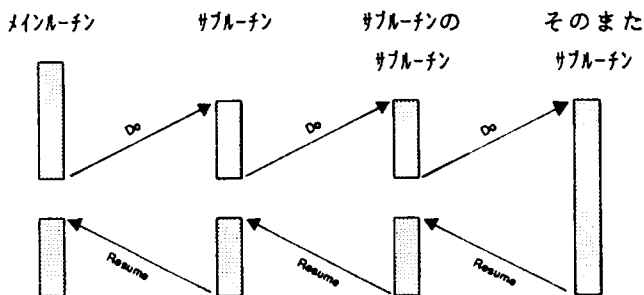
プログラムが複雑になると、作成者自身も時間がたてばその内容を忘れてしまうが、プログラムを構造化しておけばメイン・ルーチンに目を通しただけで、全体の構成を理解することができる。

反対にプログラムが“Branch to”だけでつながった１本のルーチンで書かれていると、初めから最後までコマンドをひとつひとつ追っていかなければ全容を掴むことができないという羽目になる。

★作成しやすく、修正しやすいプログラム

プログラムを書く場合、サブ・ルーチンの処理順序はメイン・ルーチンが管理してくれるので、サブ・ルーチンの配列には気を遣う必要がなくなる。

また、プログラムを修正する場合も、修正を施すルーチンがサブ・ルーチンとして括られていれば、修正の影響はそのサブ・ルーチン内に留まるが、構造化されていない１本のルーチンだと、一部の修正が時に全体の修正につながり、ともすればロジック上のエラーをひき起こす原因になりかねない。

また、下図のようにサブ・ルーチンの中にサブ・ルーチンを組み込むという手法も、プログラムの構造化、効率化を図るうえでは大切な要素のひとつである。“ＧＡＭＭＡ”のＡＩでは、探査ルーチンや攻撃ルーチンに組み込まれた移動ルーチンがこの例として挙げられる。

以上見てきた通り、“Do” と “Resume” はメイン・ルーチンとサブ・ルーチンの接続や、サブ・ルーチンとそのサブ・ルーチンの接続などルーチン間の連絡を司どるコマンドであり、一方、“Branch to” はサブ・ルーチン内部での分岐先を指示するコマンドと定義することができる。

“ＧＡＭＭＡ” の Authorize に成功したら、“ＧＡＭＭＡＳＩＭ” を Combat Simulation Module にかけて、そのパフォーマンスを観戦しよう。

SECTION 6
ＣＣＬのまとめ

SECTION BRIEF

前節の研修で、フル・カスタムデザインによるプログラムの組み方はだいたい理解していただけたと思う。この節では、プログラムの材料にあたる Cybertank Command Language (CCL) を以下の項目に沿って整理し、併せてこれまで触れられなかった部分について補足したい。

1) CCL コマンドのエレメント
2) CCL 作成パネル
3) CCL コマンドの基本形

６．１　CCL コマンドのエレメント

諸君は“ＧＡＭＭＡ”の設計を通じて、既に基本的なコマンドのほとんどに接し、それらが持つ具体的な機能について理解した。ここではこれら個々のコマンドが属するところの、概念上の分類について解説する。

６．１．１ リザーブ・ワード (Reserved Words)

CCL コマンドに使用される単語、演算記号、システム・ヴァリアブル(後述)、以上３つを合わせてリザーブ・ワードと呼ぶ。これらは全て OMEGA SYSTEM によって定義された、固有の機能を持つ因子なので、定められた定義以外の目的に使用することはできない。

※ CCL コマンドに使用される単語の全リストは p.273 に掲載

6.1.2 ラベル (Labels)

ラベルとは、ＡＩの中のあるルーチンを指す"名札"であり、"Do"コマンドや"Branch to"コマンドによって､そのルーチンを呼び出すために命名されるものである。従って、この名前はＡＩ作成者自身によって、自由に決めることができる。その他、ラベルに関する約束ごとは以下の通りである。

★ラベルは、常にＡＩウインドの左端から始まる。
★ラベルは、それだけで１行を取る。
★ラベルの文字数は最大１３文字で、英数文字、または記号を使用することができる。但し、間にスペースをはさむことはできない。
★ラベル名には、リザーブ・ワードをあてることもできる。

6.1.3 システム･ヴァリアブル (System Variables)

この概念は、いままで触れられなかった新しい概念であるが、次項のユーザー・ヴァリアブルと組み合わせて使用することによって、諸君が開発するＡＩの可能性を何倍にも拡げてくれる非常に重要な概念なので、しっかり押さえておきたい。定義は以下の通りであるが、システム・ヴァリアブルとユーザー・ヴァリアブルは対をなす概念なので、ここに併せて述べておく。

ヴァリアブルとは"変数"を意味する言葉であり、システム・ヴァリアブルもユーザー・ヴァリアブルも、ある"数値"を表わすものと理解することができる。そして、この値が何を表わすものなのか、その定義が OMEGA SYSTEM によって決定されているものを、システム・ヴァリアブル、ＡＩ作成者によって決定されるものを、ユーザー・ヴァリアブルと呼ぶのである。
抽象的な定義では理解しにくいので、まず、システム・ヴァリアブルから例を上げて解説しよう。

下線を付したのが、システム・ヴァリアブルである。

```
Detect obstruction at tank direction
If ObstacleType <= 2 then branch to GO
Fire weapon at obstruction
```

ObstacleType ： 移動センサーが探知した障害物のタイプナンバーを表わす。（p.95 参照）

```
Fire weapon at enemy tank
If ScanDamage >= 75 then branch to ESCAPE
Branch to ATTACK
```

ScanDamage ： スキャナーの現在の損傷程度を表わす。パーセンテージで管理され、0～100 の値域を持つ。

※ システム・ヴァリアブルの全リストは、p.274 に掲載

以上は、システム・ヴァリアブルの何たるかをつかんでもらうためにシステム・ヴァリアブルが、なるべく直接的な表現で使用されているコマンドを例に選んだ。が、実は“ＧＡＭＭＡ”のＡＩの中でも、既にこれが目に見えないかたちで、重要な役割を果たしているのである。次にその例を紹介しよう。

以下は、“ＧＡＭＭＡ”の攻撃ルーチンの一部である。

①==> Scan for enemy tank
②==> If enemy tank was not found then branch to Huntdone
③==> If enemy tank is beyond weapon range then branch to
        Closein
④==> Fire weapon at enemy tank

いま仮に、敵車両がスキャナーの向いている方向の射程内にいると想定して、OMEGA SYSTEM がこれらのコマンドをどのように内部で処理していくかについて見てみよう。

解説

① このコマンドでスキャナーが敵を探知すると、OMEGA SYSTEM は敵車両に関するデータを以下のシステム・ヴァリアブルにセットする。

EnemyX	:	敵車両のｘ座標
EnemyY	:	敵車両のｙ座標
EnemyDist	:	敵車両までの距離

② 敵を探知した状態なので、③の処理に移る。

③ OMEGA SYSTEM はこれにアクセスすると“EnemyDist”の値を読み込み、これが４以下であるか否かを判断する。いまは敵が射程内にいることを想定しているので、この値は４以下である。従って OMEGA SYSTEM は④の処理に移る。

④ このコマンドで OMEGA SYSTEM は “EnemyX”，“EnemyY”の値を読み込み、この座標点を攻撃する。

以上の解説をもとに、③と④をシステム・ヴァリアブルを用いたコマンドで書き表わすと以下のようになる。

③′ ==> If EnemyDist > 4 then branch to Closein
④′ ==> Fire weapon at EnemyX EnemyY

即ち、③と④のコマンドは実際にはこのようなメカニズムで処理されているのである。

これで、CCL コマンドが、システム・ヴァリアブルという“黒子”を媒介として処理されているからくりが解っていただけたと思う。
以上をまとめると、システム・ヴァリアブルとは、シミュレーションの展開を決定するあらゆる要素のなかで、OMEGA SYSTEM が“数値”として管理しているもの、と定義することができる。
そして OMEGA SYSTEM は、シミュレーションの進行と共に刻々と変化するこれらシステム・ヴァリアブルの全値を、常にメモリー上で管理し、コマンドを処理する際、いつでも必要な値を読み込めるように設計されているのである。
従って、エンジニア諸君はシステム・ヴァリアブルの種類とその定義について網羅し、シミュレーションが処理されるメカニズムを正しく理解しておくことが肝要である。

※ シミュレーション実行中、システム・ヴァリアブルの値が実際にセットされる様子や、或いは変化していく状況は、Cybertank Test Module の Status Mode を選択することによって、観察することができる。(p.154 参照)

さて、既に述べた通り、CCL はより人間の言葉に近い高級言語として開発されたものであるが、高級言語であるがゆえに、エンジニア諸君が或るコマンドを認識する際、その“言葉”から直感的にイメージする処理内容と、実際に OMEGA SYSTEM が行なう処理内容との間に微妙なズレを生じる場合がある。
既に気付いている諸君もいると思うが、前頁④のコマンドは、あくまでも、①の時点で測定した敵車両の座標位置を攻撃する内容のものであり、敵車両そのものを攻撃の対象としているわけではない。
即ち諸君が注意しなければならないのは“GAMMA”がスキャナーでフィールド上を“見る”ことができるのは、①の処理が行なわれる瞬間だけであり、②以降の行動及び判断は、全て①で得た情報だけをもとに、いわば“盲目”の状態で行なわれているということである。
従って、敵を攻撃するときは、その都度“Scan”コマンドで敵の位置をフォローしておかないと、敵が最後に測定した場所から、既に他へ移動した後も、いたずらに野を砲撃し続けるという事態に陥る。

※ 1度セットされたシステム・ヴァリアブルの値は、次にそのヴァリアブルをセットするコマンドが処理されるまで、メモリー上に保存される。

以上の例に明らかなように、CCL コマンドの機能を正確に理解するためには、そのコマンド処理がいかなるシステム・ヴァリアブルを媒介として行なわれるのか、これを押さえておくことが不可欠である。
そしてこの関係を理解することによって、初めてシステム・ヴァリアブルを使いこなすことが可能になるのである。

※ コマンドとその処理を媒介するシステム·ヴァリアブルの関係については、PART 3 の“CCL コマンド総覧”を参照のこと。

6.1.4 ユーザー・ヴァリアブル (User Variables)

システム・ヴァリアブルは、OMEGA SYSTEM によって定義された、或るファクターの値を表わすものであった。従って OMEGA SYSTEM は自動的にこれを判読して処理することができる。
これに対してユーザー・ヴァリアブルは、ＡＩ作成者が必要上勝手に定義して使用するものなので、そのＡＩ上で OMEGA SYSTEM に判読できるかたちで定義してやらなければならない。
例を上げて解説しよう。

"ＧＡＭＭＡ"の移動ルーチンは破壊不能な障害物に行き当たったときは、常に左に旋回して移動可能な方向を探るという設計だった。
これに対していま、左右を見て迂回し易いほうに旋回するというサブルーチン"BYPASS"を考えてみることにしよう。
アプローチの仕方はいろいろあると思うが、以下はその１例である。
なお、"BYPASS"は、移動ルーチンの

Turn tank left 1　　を　　Do BYPASS

と書き換えることによって、"ＧＡＭＭＡ"のＡＩのなかに組み込むことができる。（実際に書き換える必要はない）

★まず左前方４５゜の障害物を探査する。障害物が存在しないか、或いは破壊可能なものであれば、その方向にタンクの向きを合わせて元の移動ルーチンに還る。左前方４５゜の障害物が破壊不能なもの(水域を含む)であれば、次に右前方４５゜を探査する。これも破壊不能であれば、次は左右横９０゜の探査する。こうして左右の後方４５゜まで探査し、全てが破壊不能であれば、１歩後退して再びこの"BYPASS"を繰り返す。

これを CCL で組んだものが次頁のリストである。
なお、使用するシステム・ヴァリアブル、及びユーザー・ヴァリアブルの定義は次頁の冒頭に記載した通りである。

※ TankDir : 車両の向いている方角を示すシステム・ヴァリアブル
A.Dir : ユーザー・ヴァリアブル（ＡＩ上で定義）
B.Dir : 〃 （ 〃 ）

```
        BYPASS
①==>        A.Dir = TankDir
        Left1
②==>        B.Dir = A.Dir + 7
③==>        Detect obstruction at B.Dir
④==>        If ObstacleType = 2 then branch to Right1
⑤==>        If ObstacleType < 6 then branch to Face
        Right1
⑥==>        B.Dir = A.Dir + 1
⑦==>        Detect obstruction at B.Dir
            If ObstacleType = 2 then branch to Left2
            If ObstacleType < 6 then branch to Face
        Left2
            B.Dir = A.Dir + 6
            Detect obstruction at B.Dir
            If ObstacleType = 2 then branch to Right2
            If ObstacleType < 6 then branch to Face
        Right2
            B.Dir = A.Dir + 2
            Detect obstruction at B.Dir
            If ObstacleType = 2 then branch to Left3
            If ObstacleType < 6 then branch to Face
        Left3
            B.Dir = A.Dir + 5
            Detect obstruction at B.Dir
            If ObstacleType = 2 then branch to Right3
            If ObstacleType < 6 then branch to Face
```

```
Right3
    B.Dir = A.Dir + 3
    Detect obstruction at B.Dir
    If ObstacleType = 2 then branch to Back
    If ObstacleType < 6 then branch to Face
Back
    Move tank backward 1
    Branch to BYPASS
Face
    Turn tank to B.Dir
    Resume
```

解説

① これは、ユーザー・ヴァリアブル "A.Dir" を定義するためのコマンドである。即ち、"A.Dir" の定義は、OMEGA SYSTEM がこのコマンドにアクセスした時の "TankDir" の値によって決定される。
例えば、この時タンクが真北を向いていれば、"TankDir" の値は "０" なので、"A.Dir" は "０" と定義される。

② これは、ユーザー・ヴァリアブル "B.Dir" を定義するためのコマンドである。即ち、"B.Dir" は "A.Dir" から数えて、時計廻りに（４５°×７）進んだ方向を表わす値に定義される。これは "A.Dir" から見て左前方４５°に他ならない。

③ "B.Dir" の方角に障害物を探査する。

④ "ＧＡＭＭＡ" のＡＩは水陸両用のシャシーを前提としていないので、水域は避けなければならない。③で探知した障害物が水域か否か判断する。水域であれば "Right1" に分岐して別の方向を探査。水域でなければ⑤に移って、更にそれが破壊可能か否か判断する。

⑤ "ObstacleType" が６未満であれば障害物は破壊可能なので "FACE" に分岐して車両を "B.Dir" の方向に旋回させる。６以上であれば破壊不能なので、次行の "Right1" に移って別の方角を探査する。

⑥ "B.Dir" を B.Dir = A.Dir + 1 と定義し直す。

⑦ 新しく定義された "B.Dir" の方角に障害物を探査する。

以下同様にして、移動可能な方向が見つかるまで、“B.Dir” の定義を順次変えて探査を繰り返す。

ここで “A.Dir” と “B.Dir” の役割について振り返ってみよう。
“BYPASS” の設計者は、タンクが眼前の障害物を回避するためには、最悪の場合、左前方４５°から右後方４５°まで６方位探査しなければならないことを想定している。では、この６方位の指定はどのようにして行なえばよいだろうか?
それには何か基準となる方位があれば便利である。そしてその基準として最も合理的と思われるのは、障害物に突き当ったときの車両の向きである。これはその時点の “TankDir” の値に他ならない。
しかし、“TankDir” はシステム・ヴァリアブルなので、その後シミュレーションの進行と共に車両の向きが変わってしまえば、その値もそれに応じて変化してしまう。そこで必要になってくるのが、ユーザー・ヴァリアブルである。即ち、

A.Dir = TankDir

というコマンドは、この時点の “TankDir” の値を保存しておくためにＡＩ設計者が “A.Dir” という暗号を設け、これにその値を記憶させているのである。こうしておけば “A.Dir” は再定義されない限り、シミュレーションの進行に影響されることなく、いつまでも同じ値を保つことができる。そして実際に探査すべき方角は、別にユーザー・ヴァリアブルを立て、“A.Dir” を基準にして計算式で定義してやればよい。これを実行しているのが

B.Dir = A.Dir + 数値

のコマンドである。

以上に明らかなように、ユーザー・ヴァリアブルの定義とは、ＡＩプログラムのなかで使用する“或る値”を、必要なときにいつでも呼び出せるよう、ＡＩ作成者が命名した暗号に、その値を記憶させるという行為を指すのである。

なお、以下はユーザー・ヴァリアブルの使用に関する約束ごとである。

★ユーザー・ヴァリアブルの文字数は、最大１３文字で、英数文字や記号を使用することができる。但し、スペースをはさむことはできない。

★ユーザー・ヴァリアブルに、リザーブ・ワードを単独であてることはできないが、その一部として含むことはできる。
例えば、“Scan”はユーザー・ヴァリアブルとして使用することはできないが、“EnemyScan”は使用することができる。

★ユーザー・ヴァリアブルの値域は、０～１００である。

※ アクション・コマンド(次頁参照)の数値設定に、ユーザー・ヴァリアブルを用いる際、(＋),(－)を含んだ式を直接代入することはできない。

(例)　Detect obstruction at B.dir　(○)
　　　Detect obstruction at A.dir + 1　(×)

６．１．５ 設定コマンド (Assignment Commands)

設定コマンドは、ユーザー・ヴァリアブルの値を定義（設定）するためのコマンドで、以下３つの形式がある。但し、（－）の前には必ずスペースを入れなければならない。（＝）（＋）の前後はスペースを入れても入れなくてもよい。

<U.Var> = <Var/Val>
<U.Var> = <Var> + <Var/Val>
<U.Var> = <Var> - <Var/Val>

<U.Var> ： User Variable
<Var> ： User Variable or System Variable
<Var/Val> ： User Variable , System Variable or Value (数値)

> ※ ユーザー・ヴァリアブルの値域は０～１００である。
> １００を越える数が設定された場合、ユーザー・ヴァリアブルの値は自動的に“１００”に設定され、マイナスの数が設定された場合は自動的に“０”に設定される。

６．１．６ アクション・コマンド (Action Commands)

アクション・コマンドとは、サイバータンクに物理的な運動、即ち機械的な動作を命じるもので、下に列記したものの他に、特殊アイテムを操作するコマンドなどがこれに含まれる。

Move
Turn
Detect
Scan
Rotate
Fire

6.1.7 シーケンス・コマンド (Sequence Commands)

コマンドの処理は通常、1番上の行からその配列順に行なわれるが、この処理順序を変更するために使用するのが、シーケンス・コマンドである。シーケンス・コマンドには以下の3種類がある。

```
Branch to   (= Goto )
Do          (= Gosub)
Resume
```

★ Branch to と Do は、後に必ずラベル名を伴わなければならない。
★ Branch to と Do は、それぞれ "Goto" と "Gosub" で代用することができる。

6.1.8 判定コマンド (Decision Commands)

判定コマンドは、別名 "If コマンド" とも呼び、以下の構造を持つ。但し、<seq.com> は "resume" を除くシーケンス・コマンド。

If ＜ 判定条件 ＞ then ＜ seq.com ＞

★ OMEGA SYSTEM は If コマンドにアクセスすると、まず判定条件をチェックし、これが満たされている場合はシーケンス・コマンドを処理し、満たされてない場合はこれを無視して次行の処理に移る。
★ then の後に、"branch to" 或いは "Do" 以外のコマンドを置くことはできない。

(例) If movement is obstructed then turn tank left 1 (×)
　　 If enemy tank was not found then resume (×)

また、判定条件にユーザー・ヴァリアブルが含まれる場合、その形式は以下の３つに分類される。

```
If    <Var> = <Value>              then   <seq.com>
If    <Var> = <Var> + <Value>      then   <seq.com>
If    <Var> = <Var> - <Value>      then   <seq.com>
```

判定条件は、等号(＝)の部分を以下の関係演算子に置き換えることによって、さまざまな大小関係を判断することができる。

<	（未　満）
>	（より大）
<＝	（以　下）
>＝	（以　上）
<>	（不等号）

６．１．９　ロジック・コマンド（Logic Commands）

これは、アクション・コマンドに対する概念であり、サイバータンクの機械的な動きを伴わない、コンピュータ上の計算処理のみで完了するコマンドである。上述の

設定コマンド
シーケンス・コマンド
判定コマンド

がこれに属する。

6.1.10 サイクル・カウント (Cycle Counts)

CCL コマンドは、全てその処理に一定の時間を要し、その処理時間はサイクル (cycle) と呼ばれるものを単位として測定される。
CCL は基本的にロジック・コマンドとアクション・コマンドの２種類に分かれるが、サイクル数との関係は以下の通りである。

1) ロジック・コマンドは、サイバータンクの機械的な動きを伴わない、コンピュータ上の計算処理だけなので、その処理時間は極めて短い。全てのロジック・コマンドの処理時間は 1 cycle である。
2) アクション・コマンドは、その種類によって処理時間に幅がある。主なものを下に列記する。

Detect obstruction at tank direction	4 cycle
Scan for enemy tank	8 cycle
Turn tank left 1	16 cycle
Rotate Scanner left 1	16 cycle
Move tank forward 1	40 cycle
Fire weapon at enemy tank	40 cycle

ここに記したサイクル数は、全てそのアクション・コマンドの単位当たりのサイクル数である。従って、 Move tank forward 3 というコマンドの処理には (40×3) cycle の時間を要する。

この概念はここで初めて紹介するものであるが、実は、このサイクルカウントというのは、これをどれだけ短縮することができるかによって、勝敗の行方を大きく左右してしまう、サイバータンクにとっては死活的な要素と言うことができる。

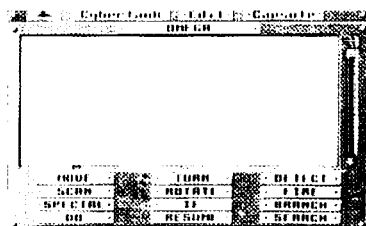
例えば、諸君が必ず先制攻撃を加えることができるようなＡＩを設計し、武器の破壊力がほぼ同等の敵と対戦させたとしてみよう。ところが運悪く、相手の武器はレーザー系で、１回の攻撃に要するサイクル数は諸君らのそれの１/２であったとする。
これではいくら先制攻撃を加えたところで、１発お見舞いする間に２発食らう勘定になり、諸君らに勝ち目はない。

これは、極端な例ではあるが、ＡＩの性能を正確に評価するためには、ＡＩ回路の周到さのみならず、“処理時間”という、ＡＩが持つ別の側面にも留意しなければならないということをよく言い表わしている。そして、常にこのことを意識してプログラムを組むことが贅肉のない、強力なＡＩの設計につながるのである。

※ 他のコマンドのサイクル数については、PART 3 参照のこと。

6.2　CCL作成パネル

CCL 作成パネルは、CCL コマンドを入力する際、キーボードからいちいちタイピングしなくても、単語の書かれたパネルを選択していくだけでコマンドの構文を作成してくれる、非常に便利な機能である。
以下に、その主な利点を記す。

★タイプ入力の手間が省けるので、プログラミングの効率が上がる。
★タイプ入力に起因するエラー(誤字、脱字など)を防ぐことができる。
★一部の特殊なコマンドを除けば、OSI によって認定されたほとんど全てのコマンドを作成することができる。

6.2.1 CCL作成パネルの使い方

では、実際にパネルを使って、いくつかの典型的なコマンドを作成してみよう。

> ※ 作業は既存のタンクのＡＩを呼び出し、その余白を利用して実際に試してみること。なお作業終了後 AI Module を出る際は、“Save changes to *** ?” に対して “No” を選択する。

★ "Move tank forward 1" の作成

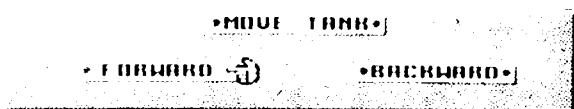
1) まず初めに、コマンドを挿入する場所にカーソルを合わせる。

2) パネル [MOVE] を選択する。

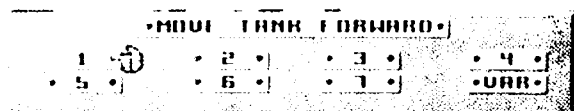
(キーボードの場合) [ESC] キーで ☜をパネルの欄に降ろし、更に [←],[→],[↑],[↓]キーで、☜を [MOVE] に合わせて、[↵]キーを押す。

(マウスの場合) [MOVE] を直接クリックする。

3) [FORWARD] を選択する。

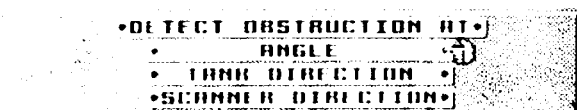
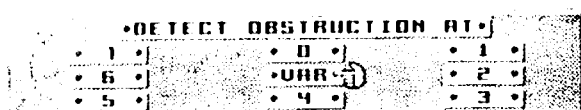
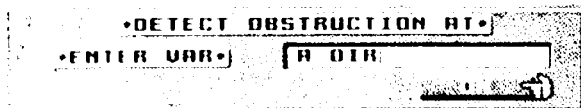
初期のパネル・メニューに戻りたい場合は、パネル・ボード最上行の "センテンス・パネル" を選ぶ。(画面の [MOVE TANK])

4) [1] を選択する。

前の画面に戻るときは、同じくセンテンス・パネルを選ぶ。

(画面の [MOVE TANK FORWARD])

※ この機能で入力したコマンドは、[Undo] 機能で削除することができる。

★ "Detect obstruction at A.Dir" の作成

1) DETECT を選択する。

2) ANGLE（方位）を選択する。

3) VAR（Variable）を選択する。
これは、システム・ヴァリアブルやユーザー・ヴァリアブルを入力するときに選ぶパネルである。なお、0～7の数字は絶対方位を表わすもので、パネル・ボード上の配置もその方角に合わせて配置されている。

4) キーボードから "A.Dir" と入力し、OK を選択する。

★ "If movement is not obstructed then branch to Go" の作成

1) [IF] を選択する。

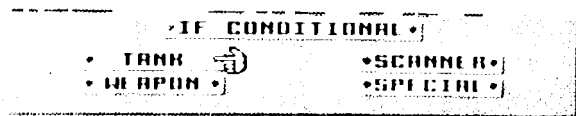
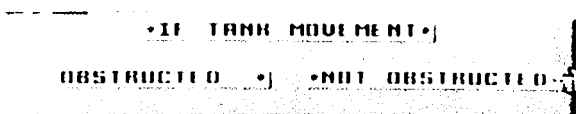
2) [TANK] を選択する。

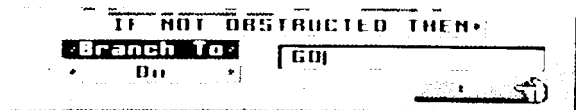
```
                 IF CONDITIONAL
  TANK                          SCANNER
  WEAPON                        SPECIAL
```

3) [MOVEMENT] を選択する。

```
                 IF TANK
  TREADS                        MOVEMENT
  ALIGNMENT                     FUEL
```

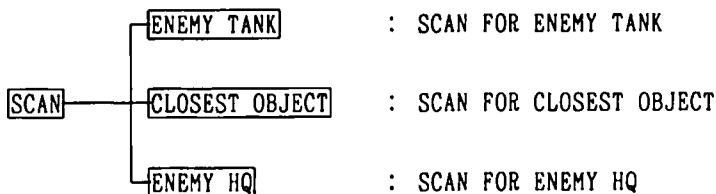
4) [NOT OBSTRUCTED] を選択する。

```
            IF TANK MOVEMENT
  OBSTRUCTED          NOT OBSTRUCTED
```

5) Ifコマンドは必ずシーケンス・コマンドを伴うので、ここで分岐先のラベル名 "GO" をキーボードから入力する。

```
       IF NOT OBSTRUCTED THEN
  Branch To      GO
  Do
```

※ シーケンス・コマンドが "Do" の場合は [DO] を選択してこれを反転させ、入力が済んだら [OK] を選択する。

CCL 作成パネルの持つ利点は、入力における効率ばかりではない。
初めにも述べた通り、CCL 作成パネルは、ほとんど全てのコマンドを作成することができるので、或るコマンドのヴァリエーションについて知りたいと思ったときは、同名のパネルをめくり、それに続く選択肢の全ての組み合わせについて見てみればよい。

例えば SCAN パネルを開いてみるとわかるが、このパネルが持つ選択肢の組み合わせは、下の樹系図が示すように3通りしかない。従って OSI が認定した "Scan" コマンドのヴァリエーションも、以下の3種類しかないということが判る。

```
         ┌[ENEMY TANK]        : SCAN FOR ENEMY TANK
[SCAN]───┼[CLOSEST OBJECT]    : SCAN FOR CLOSEST OBJECT
         └[ENEMY HQ]          : SCAN FOR ENEMY HQ
```

即ち、逆を言えば、CCL 作成パネルで書くことのできないコマンドはほとんどイリーガルなものであるということが言えるので、例えば、"Scan for obstruction" という概念的には不自然でないコマンドも、SCAN パネルのヴァリエーションを覗いてみることによって、これが無効であることが判る。

このように、CCL 作成パネルは、コマンドの種類や文型に対する理解をより確実にするうえでも、役立つ機能なのである。

6.2.2 パネル“SEARCH”の使い方

パネル・メニュー右下の SEARCH パネルは、AI上の或る特定のテキスト(文字、単語、文節など)を検索するときに使用する便利な機能であり、また検索したものを別のテキストに書き換えることもできる。なお、検索可能なテキストの長さは、スペースも含めて12文字までである。

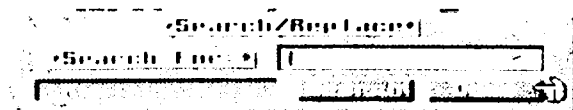
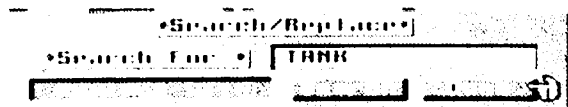
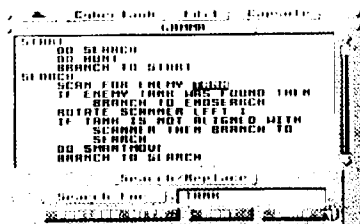
★テキストの検索

例えば、Authorize したときに、以下のようなエラー・メッセージが出たとする。

'ENEMYTANK' does not compute in the line :
SCAN FOR ENEMYTANK

これは、ENEMY と TANK の間にスペースを入れなかった単純なミスであるが、AIが長く複雑なもので、“ SCAN FOR ENEMY TANK ” というコマンドがいろいろな場所で使われている場合、その行を見つけ出すのに苦労する場合がある。
こんなとき、SEARCH パネルを使えば、指定したテキストを即座に捜し出して、これを反転表示してくれるのである。

では、実際に“GAMMA”のAIを使って“tank”というテキストを検索してみよう。

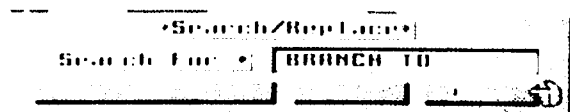
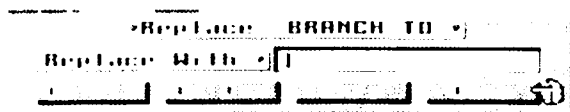
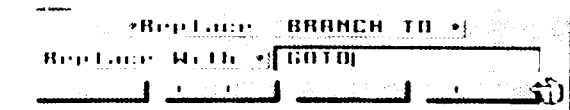
1) 検索はカーソルのある位置から下に向かって行なわれるので、初めにカーソルをAIの冒頭に戻しておく(HOME キーが便利)。
2) SEARCH を選択すると、ボードが以下のように切り換わる。

3) キーボードから“tank”と入力する。

4) Search を選択すると、最初に検索した“tank”を反転表示する。

5) Next を選択すると、次の“tank”を反転表示する。これを繰り返していくと、一通り検索を終えた時点でビープ音が鳴る。
ここでもう一度初めから検索したい場合は、再び Next を選ぶ。
検索を終了するときは、Done を選択する。

★テキストの置き換え

次にこの機能を使って“branch to”の部分を“goto”に書き換えてみよう。

1) SEARCH を選択したら、“branch to”と入力する。

※テキストを入力するウィンドの中には、前回検索したテキストが反転表示されているが、これは BS キー、もしくは DEL キーを1回押すだけで、クリアすることができる。

2) Search を選択する。

3) 次に Replace With を選ぶと、パネル・ボードが下のように切り換わる。

4) "goto" と入力する(go と to の間にスペースを入れないこと)。

5) ボード上の4つのパネルは、それぞれ以下の機能を持つ。

Do All ……… すべての "branch to" を "goto" に書き換える。

Do One ……… 現在反転している "branch to" だけを "goto" に書き換える。カーソルは自動的に次の "branch to" を反転表示する。

Skip ………… 現在反転している "branch to" は書き換えずそのままにして、次の "branch to" を反転表示する。

Done ………… 検索を終了する。

ここでは、Do One と Skip を使って、"branch to" のいくつかを "goto" に書き換えてみよう。なお、"goto" は "branch to" と全く同じ機能を持つので (p.221 参照)、このまま Authorize しても、プログラムの内容に変更はない。

6.2.3 “Expanded Text” について

CCL 作成パネルによるコマンド表記の仕方には２種類のものがあり、諸君がこれまで目にしてきたのは全て “Expanded Text” (拡張形) と呼ばれるものである。この “Expanded Text” が、より自然な英語に近い構造を持っているのに対し、もう一方の “標準形” と呼ばれるものは、OMEGA SYSTEM が解読できる必要最小限の語彙で構成されている。初期設定は、“Expanded Text” モードになっており、これを標準形のモードに切り替えるためには、[Edit] の [Expanded text] を選択してこれを解除する。(解除すると [Expanded text] の頭のチェック・マークがとれる)

以下は、例として先ほど作成した３つのコマンドを標準形で表記したものである。

```
Move  tank  forward  1                              (拡張形)
Move  forward  1                                    (標準形)

Detect  obstruction  at  A.Dir                      (拡張形)
Detect  at  A.Dir                                   (標準形)

If movement is not obstructed then branch to Go     (拡張形)
If  not  obstructed  then  Go                       (標準形)
```

※ この例からわかるように、If コマンドの “then” の後の “branch to” は省略可能である。但し、“Do” は省略不可。
※ コマンド内の省略可能な単語に関しては、PART 3 の CCL コマンド総覧を参照のこと。

6.3　CCLコマンドの基本構文

より強力なＡＩの開発を任務とする諸君は、言うまでもなく、全てのコマンド・ヴァリエーションについて理解していなければならない。
そこで、CCL 作成パネルのボード上の配列に沿って、各コマンドのヴァリエーションについて見ていくことにしよう。
なお、各コマンドを作成するためのパネルの選択手順に関する説明は省略したので、主だったものについては、AI Module 上で実際に試行錯誤しながら作成してみること。これが、CCL 作成パネルを使いこなすための近道である。

> ※ CCL 作成パネルによって作成できないコマンドの一覧については、P.152 参照のこと。

6.3.1　“Move” コマンド

★ “Move” コマンドのヴァリエーションはこの１種類のみで、移動距離の設定は、確定値(値域 0～63)かヴァリアブルによって行なう。

Move	Tank	forward	0～63
		backward	Var

> ※ Var ： System Variables　or　User Variables

6.3.2 “Turn” コマンド

★現在タンクの向いている方向を基準にして、左右に何度方向転換させるかを指示するコマンド。

```
Turn  tank    left     0～7
              right    Var
```

★現在の車両の向きに関わらず、方向転換すべき方向を絶対方位によって、指定するコマンド。

```
Turn  tank  to   0～7
                 Var
```

★指定した座標点(x：x座標，y：y座標)の方向に車両を方向転換させるコマンド。

```
Turn  tank  to   x   y
```

・xとyの値域は 0～63 (Battlefield 外枠の座標も含む)。
・xとyは、ヴァリアブルによって指定することもできる。
・入力の際は、“x”と“y”の間にスペースを入れる(，は不可)。
なお、座標点のある方向が車両が方向転換できる8方位(絶対方位)の中間に位置した場合は､OMAGA SYSTEM がそれに最も近い絶対方位を算出し、その向きに車両を方向転換させる。

★最後に“Scan”した目標物の座標に車両の向きを合わせるコマンド。
なお、目標物が絶対方位の中間に位置した場合は、それに最も近い方位に車両を方向転換させる。

Turn tank to face enemy tank
enemy HQ

(※ Turn tank to face closest object は不可)

★スキャナーの向いている方向に車両の向きを合わせるコマンド。

Align tank with scanner

6.3.3 “Detect”コマンド

★移動センサーの探査すべき方向を、絶対方位で指示するコマンド。

Detect obstruction at 0〜7
Var

★移動センサーの探査すべき方向を、特定の向きに合わせるコマンド。

Detect obstruction at tank direction
scanner

※“Detect”コマンドによってセットされるシステム・ヴァリアブルの種類については、p.197を参照のこと。

6.3.4 "Scan" コマンド

★特定の目標物を探査するコマンド

```
Scan  for   enemy tank
            enemy HQ            (敵指令部)
            closest object      (最も近くにある物体)
```

※ スキャナーは Battlefield の構成要素のうち、タイプ・ナンバー3以上を "closest object" として認識し得る。(p.95 参照)

※ これら3つのコマンドによってセットされるシステム･ヴァリアブルの種類については、p.200〜202 を参照のこと。

6.3.5 "Rotate" コマンド

★現在スキャナーの向いている方向を基準にして、スキャナーを左右に何度旋回させるかを指示するコマンド。

```
Rotate  scanner  left    0〜7
                 right   Var
```

★現在のスキャナーの向きに関わらず、スキャナーを旋回させる方向を絶対方位によって指定するコマンド。

```
Rotate  scanner  to  0〜7
                     Var
```

★指定した座標点(x：x座標，y：y座標)の方向にスキャナーを旋回させるコマンド。

Rotate scanner to x y

・xとyの値域は 0～63 (Battlefield 外枠の座標も含む)。
・xとyは、ヴァリアブルによって指定することもできる。
・入力の際は、xとyの間にスペースを入れる(, は不可)。
なお、座標点の方向が絶対方位の中間に位置した場合は、それに最も近い方位にスキャナーを旋回させる。

★最後に“Scan”した目標物の座標に、スキャナーの向きを合わせるコマンド。なお、目標物が絶対方位の中間に位置した場合は、それに最も近い方位にスキャナーを旋回させる。

Rotate scanner to face enemy tank

(※ Rotate scanner to face enemy HQ は不可)

★スキャナーの向きを車両の向きに合わせるコマンド。

Align scanner with tank

6.3.6 "Fire" コマンド

★移動センサーが最後に探知した障害物の座標点を攻撃するコマンド。

```
Fire  weapon  at  obstruction
```

★スキャナーが最後に探知した目標物の座標点を攻撃するコマンド。
但し、目標物が射程外の場合は、その方向に向けて砲撃する。

```
Fire  weapon  at    enemy tank
                    enemy HQ
                    closest object
```

★或る特定の方角を攻撃するコマンド。

```
Fire  weapon  at    tank        direction
                    scanner
```

★指定した座標点(x：x座標，y：y座標)を攻撃するコマンド。
但し、座標点が射程外の場合は、その方向に向けて砲撃する。

```
Fire  weapon  at  x  y
```

・xとyの値域は 0〜63 (Battlefield 外枠の座標も含む)。
・xとyは、ヴァリアブルによって指定することもできる。
・入力の際は、xとyの間にスペースを入れる(，は不可)。

6.3.7 “Special” コマンド

★ 万策尽きたときに自爆するコマンド（p.214 参照）

Self Destruct

★ リモート・スキャナーを発射するコマンド（p.207 参照）

Launch Remote Scanner

★ シールドの上げ下げを行なうコマンド（p.213 参照）

Raise Shield
Lower Shield

★ 敵のスキャナー・ロックを解除するコマンド（p.206 参照）

Jam Scanner Signal

★ 敵に対しスキャナーをロック／解除するコマンド（p.204 参照）

Lock Scanner
Unlock Scanner

★ 損傷した部位を修理するコマンド（p.211 参照）

```
Repair   Internal
         Treads
         Weapon
         Armor
         Scanner
```

6.3.8 “If” コマンド

★ 車両の移動が妨げられているか否かを判断するコマンド

```
If  movement  is (not)  obstructed  then  <seq.com>
```

★ スキャナーが目標物を探知したか否かを判断するコマンド

```
If   enemy tank       was (not) found  then  <seq.com>
     enemy HQ
     closest object
```

★ 攻撃対象が射程内であるか否かを判断するコマンド

```
If   enemy tank      is   within   weapon  range  then
     enemy HQ             beyond                <seq.com>
     closest object
```

★ 特定の部位が機能しているか否かを判断するコマンド

```
If    tank treads   are   (not) functional then <seq.com>
      weapon        is
      scanner       is
```

★ 車両が目標物の方向を向いているか否かを判断するコマンド

```
If tank is (not) facing   enemy tank    then   <seq.com>
                          enemy HQ
```

★ 車両の向きとスキャナーの向きが一致しているか否かを判断するコマンド

```
  If  tank    is (not) aligned  with  scanner  then <seq.com>
= If scanner is (not) aligned  with  tank     then <seq.com>
```

★ 敵をスキャナー・ロックしているか否かを判断するコマンド

```
If  scanner  is    locked       then  <seq.com>
                   unlocked
```

★ シールドが上がった状態であるか否かを判断するコマンド

```
If  shield  is  up        then  <seq.com>
                down
```

★ リモートスキャナー/修理用キットが使用できるか否かを判断するコマンド

```
If  remote scanner  is  available    then  <seq.com>
    repair kit          unavailable
```

6.3.9 "Branch" "Do" "Resume" コマンド

★ 分岐先のラベル名を指示するコマンド

```
Branch  to  "ラベル名"
```

★ サブ・ルーチンの実行を指示するコマンド

```
Do  "ラベル名"
```

★ 上位ルーチンへの復帰を指示するコマンド

```
Resume
```

＊＊＊　ＣＣＬ作成パネルで作成できないコマンド一覧　＊＊＊

If obstruction is ally (enemy) HQ then <seq.com> (p.197)

If tank is (not) facing "X" "Y" then <seq.com> (p.198)

If closest object is ally (enemy) HQ then <seq.com> (p.202)

If tank is (not) being scanned then <seq.com> (p.205)

Get Distance to "X" "Y" (p.215)

Get Random (to #) (p.216)

Beep (p.217)

* (p.218)

Break (p.219)

If Last Key pressed then <seq.com> (p.220)
If Last Key pressed = # then <seq.com>

Include "Label" (p.222)

Switch Commlink on (off) (p.223)
Transmit #
Clear Data
Copy Data

及び、条件判定にヴァリアブルを含む"Ｉｆコマンド"

SECTION 7
ＡＩプログラムのデバッグ

SECTION BRIEF
この節では作成したサイバータンクの性能を詳細に分析する際、特にＡＩプログラムの“デバッグ”（プログラム上の誤りを検出すること）を行なう上でエンジニアにとって無くてはならない基本ツール――Cybertank Test Module について解説する。

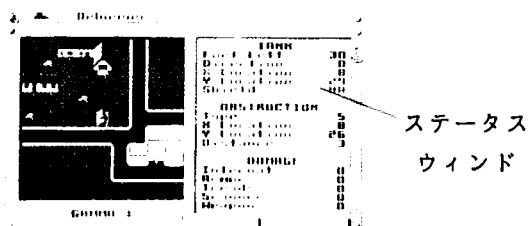
7．1　Cybertank Test Module

エンジニアの間から“デバッガー”という愛称で親しまれているこのツールは、“Status Mode”と“Trace Mode”と呼ばれる２つの機能から構成されている。その具体的役割は、戦闘シミュレーションとこれを処理する際に読み込まれるプログラムやシステム・ヴァリアブルの値を画面に並走させ、この対比によってサイバータンクの持つさまざまな機能やＡＩプログラムの問題点を詳細に分析することにある。
１つのタンクをこのモジュールにかけて分析するためには、まずそのタンクを Primary Tank とするシミュレーション・デザイン・ファイルを作成しなければならない。ここではとりあえず“ＧＡＭＭＡＳＩＭ”を使って以下の解説を見ていこう。

・ECM に戻ったらメニュー [Design] の [Test Cybertank] を選択して、“ＧＡＭＭＡＳＩＭ”を選ぶ。
・戦闘が始まったら、ひとまず [CTRL] +Ｐでポーズをかける。
（ポーズ解除は再度 [CTRL] +Ｐ）

Cybertank Test Module アクセス時の初期モードは“Status Mode”になっているので、このモードの解説から始めよう。

7.1.1 "Status Mode"

このモードは、 OMEGA SYSTEM が Primary Tank のＡＩを処理する際に使用する"ステータス値"（システム·ヴァリアブルやユーザー·ヴァリアブルの値など）を画面に表示するもので、エンジニアはこれを見ることによって現在のタンクの状態や、そのタンクの外界に対する認識状況を確認することができる。
下の画面を例にして、具体的に解説しよう。

Cybertank Test Module (Status Mode)

解説

★ ステータス·ウィンドウ内の項目は、システム·ヴァリアブルに相当するもので、右側の数字がその値を示している。個々の項目がどのシステム·ヴァリアブルに対応するかは、p.158〜162 を参照。

★ 上の画面から、現在"ＧＡＭＭＡ"が（ｘ，ｙ）＝（ 8,29）の位置にいて絶対方位"０"の方角を向き、（ｘ，ｙ）＝（ 8,26）の位置にタイプ・ナンバー"５"の障害物を探知した、という情報などが得られる。

★ 照会できるシステム・ヴァリアブルは全部で４８種類である。
以下の操作でステータス・ウィンドの頁をめくれば、その全てを見ることができる。
（キーボードの場合）[←],[→],[↑],[↓]キーで☞を [Next](次頁)、或いは[Prev](前頁)に合わせて[↵]キーを押す。
（マウスの場合）[Next], [Prev] をクリック。
この操作は">"キー(次頁)と"<"キー(前頁)で代用できる。

★ ステータス・ウィンドの最終頁には、ユーザー・ヴァリアブルとその値が表示される。（但し、Primary Tank のみ）
例えば、“ＧＡＭＭＡ”の移動ルーチンに p.124 の “BYPASS” が組み込まれている場合、この頁に “A.Dir” “B.Dir” 及びその値が表示される。

★ ポーズ、及びポーズ解除は CTRL +Ｐを押す。

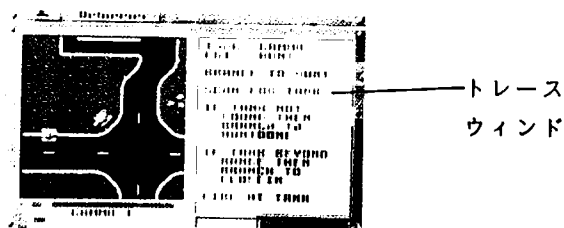
★ タンク･セレクションキー(p.64 参照)を押すことによって他車両のステータス値を見ることができる。

★ 車両の位置や状態を表わすステータス値は、これを変更することによって、各車両とも任意の状況を設定することができる。

1) ポーズをかける。（かけなくても可）
2) タンク･セレクションキーを押してステータス値を変更する車両を選択する。
3) 変更するステータス値を反転させる。
   (キーボードの場合) 変更する値に☞を合わせ⏎キーを押す。
   (マウスの場合) 変更する値をクリック。
4) キーボードから新しい値を入力し、⏎キーを押す。
   このとき、ステータス値(ヴァリアブル)の値域を越える値を入力すると、自動的にその最大値がセットされる。
5) ポーズを解除する。

７．１．２ “Trace Mode”

フル・カスタムデザインで複雑なＡＩを組むようになると、タンクが自分の意図通りに動いてくれなかったり、訳のわからぬ行動をとったりする場面にしばしば突き当る。このような事態は勿論エンジニアの不注意やコマンドに対する認識不足で、実際にはプログラムが意図通り組まれていないために生じるのだが、リストを見返すだけではその原因を解明するのに苦労する場合がある。
以下に解説する “Trace Mode” は正にこのような局面で、車両の行動とそれを動かしているプログラムを画面上に同時進行させ、この対比によってプログラムの問題点を徹底的に分析する最強のデバッグ・ツールである。

“Trace Mode” に切り替えるには、Debugger の Trace Mode On を選択する。(“Status Mode” に戻るときは再度 Trace Mode On を選ぶ)

Cybertank Test Module (Trace Mode)

解説

★ トレース・ウィンド内には、処理中のＡＩプログラムが下から上にスクロール表示され、最下行に現われたコマンドがまさに現在処理中のコマンドであることを表わす。

★ トレース・ウィンド最上部には、タンク名と現在処理中のラベル名が表示される。

★ シミュレーション実行中、以下の操作によってコマンドの処理を１ステップずつ手動で進ませ、１つひとつのコマンドがどのような効果をもたらすかを確認することができる。

1) Debugger の Single Step On を選んで処理を一時停止する。これはこのモジュールのポーズ機能でもあり、CTRL +P で代用することがでる。（ポーズ解除は再度 Single Step On を選択するか、CTRL +P を押す)

2) SPACE キーを１回押すごとに１コマンドずつ処理が進行する。

★ “Trace Mode” では、Primary Tank 以外のＡＩプログラムを照会することはできない。

★ このモジュールを抜けるときは、Debugger の Exit を選択する。

※ Cybertank Test Modüle を使用するとき、Primary Tank として選択できるのは、エンジニアが自分で Authorize した実行ファイルに限られる。他のディレクトリからファイル・コピーした実行ファイルは、このモジュールでロードすることはできない。
(ファイル・コピーについては SECTION 9 を参照のこと)

＊＊＊ ステータス一覧 ＊＊＊

p. 1

TANK

Fuel Left	FuelLevel
Direction	TankDir
X Location	TankX
Y Location	TankY
Shield	※

OBSTRUCTION

Type	ObstacleType
X Location	ObstacleX
Y Location	ObstacleY
Distance	ObstacleDist

DAMAGE

Internal	IntDamage
Armor	ArmorDamage
Treads	TreadDamage
Scanner	ScanDamage
Weapon	WeapDamage

※ Shield : Shield を装備していないとき NA 、
装備していてこれを上げているとき UP 、
装備していてこれを下げているとき DN と表示される。

p. 2

SCANNER

Locked	※
Direction	ScanDir

Enemy Tank

X Location	EnemyX
Y Location	EnemyY
Distance	EnemyDist

Object

Type	ObjType
X Location	ObjX
Y Location	ObjY
Distance	ObjDist

Enemy HQ

X Location	EnemyHQX
Y Location	EnemyHQY
Distance	EnemyHQDist

※ Locked : Scanner Lock を装備していないとき NA 、
装備していて敵をロックしているとき YES 、
装備していて敵をロックしてないとき NO と表示される。

p.3

COMMLINK DATA

Ally Number	AllyNum
Code Received	AllyCode

ALLY TANK

X Location	AllyX
Y Location	AllyY
Distance	AllyDist

ENEMY TANK

X Location	AllyEnemyX
Y Location	AllyEnemyY
Distance	AllyEnemyDist
Direction	AllyEnemyDir

p.4

COMMLINK COPY

Ally Number	CopyNum
Code Received	CopyCode

ALLY TANK

X Location	CopyX
Y Location	CopyY
Distance	CopyDist

ENEMY TANK

X Location	CopyEnemyX
Y Location	CopyEnemyY
Distance	CopyEnemyDist
Direction	CopyEnemyDir

p.5

MISCELLANEOUS（その他）

Scanned?	※
Repair Kits	KitsLeft
Launchers	RemotesLeft
XY Distance	XYDist
Random #	RandomNum

ALLY HQ

X Location	AllyHQX
Y Location	AllyHQY

p.6

USER VARIABLES

この欄にはＡＩプログラムに組み込まれている全てのユーザー・ヴァリアブルとその値が表示される。

※ Scanned? : Listener を装備していないとき NA 、装備していて敵にスキャナー･ロックされているとき YES 、装備していて敵にスキャナー･ロックされていないとき NO と表示される。

SECTION 8
等級評価

SECTION BRIEF
この節では、等級評価を行なう Clearance Evaluation Module について解説する。

8.1 等級評価

等級評価とは、サイバータンク・エンジニアの質的向上を図るために OSI が採用している昇級制度である。
評価の具体的方策は、エンジニアの設計したタンクと、各等級ごとに用意された OSI のタンクとの間で戦闘シミュレーションを行ない、その戦績によって昇級の可否を決するというものである。
戦闘シミュレーションは１０戦行なわれ、昇級を果たすためにはそのうち７勝以上を修めなければならない。
昇級による具体的な利益は予算の増額であり、１ランク昇級するごとにタンク１両の設計にかけられる予算が 1,０００ Credits ずつ加算される。
なお、等級評価実行中は、ポーズをかけることはできない。
等級評価を受けるための操作は以下の通りである。

1) ECM に戻り、メニュー Employee の Clearance Evaluation を選択する。

2) ファイル・ウィンドが開いたら、評価にかけるタンクのファイルを反転させ、最後に Open を選択する。

***** 等級名一覧 *****

ランキング	等級名	予算額(CREDITS)
1	STANDARD	１０００
2	CONFIDENTIAL	２０００
3	SENSITIVE	３０００
4	RESTRICTED	４０００
5	CLASSIFIED	５０００
6	PRIVILEGED	６０００
7	SECRET	７０００
8	TOP SECRET	８０００
9	EYES ONLY	９０００
10	OMEGA	無 制 限

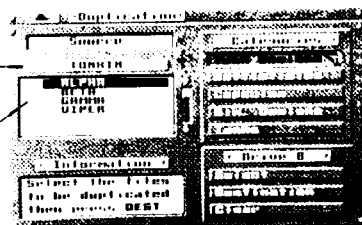
SECTION 9
ファイルのコピー

SECTION BRIEF

諸君にとって当面の目標は、タンクの設計に改善を重ね、等級評価の最高ランキングを勝ち取ることであるが、この目標は単なる通過点にすぎず、これを達成した後もエンジニアとしての研鑽は積んでいかなければならない。その具体的方策として最も適切と思われるのがエンジニア同士での対戦である。
この節では、エンジニアが作成した車両同士の対戦を実現するために不可欠な、ディスク間のファイル・コピーを行なう Data Duplication Module の機能について解説する。

9.1 Data Duplication Module

OMEGA SYSTEM が、戦闘シミュレーションを処理するためには、シミュレーション・デザイン・ファイルを構成する3つの要素、即ち自車両、敵車両、及び戦場の各ファイルが同一ディスク上になければならないという条件がある。
従って、諸君が他のエンジニアが作成したサイバータンクを敵車両に選んで戦闘シミュレーションを行なうためには、そのファイルを諸君のＩＤディスク上にファイル・コピーしなければならない。これを行なうのが Data Duplication Module である。

では、ECM からメニュー Design の Duplication Module を選択して Data Duplication Module にアクセスしよう。

Data Duplication Module

解説

★ Categories の欄にある５つの項目は、それぞれ次のようにコピーするファイルのカテゴリーを表わす。

Tank Designs ………… サイバータンクのソース・ファイル
Battlefields ………… 戦場ファイル
Capsules ………… カプセル・ルーチンのファイル
Sim.Designs ………… シミュレーション・デザイン・ファイル
Tanks ………… サイバータンクの実行ファイル

★ この５つのカテゴリーは選択されるとその表示が赤に変わり、ファイル・ウィンド内にそこに納められたファイルが表示される。

★ "ディレクトリ表示パネル" には、アクセス中のディレクトリ名が表示される。

★ "ドライブ指定パネル" は、カレント・ドライブを変更するときに使用する。即ち、OMEGA SYSTEM がアクセスすべきドライブを指定するときに用いる。（後で具体例で解説する）

★ Information の欄には状況に応じて指示が出される。

では、実際の操作手順を具体例をあげて解説しよう。

★　2ドライブ・マシーンの場合

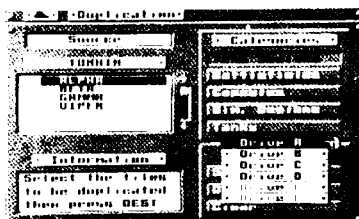
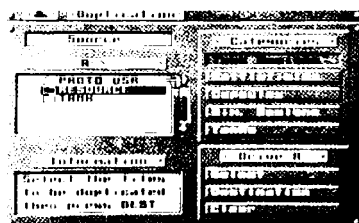
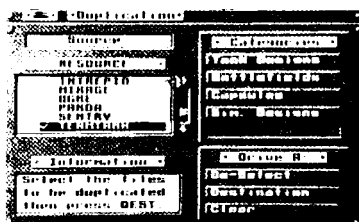
いま、Aドライブに“OMEGA”のシステム・ディスク、Bドライブに I Dディスクがセットされていると仮定し、システム・ディスクのディレクトリ“RESOURCE”から、実行ファイルの“TEAM TANK”をIDディスクのディレクトリ“TONKIN”にコピーしてみよう。

1) まず、複写元のディスクと複写先のディスクを両ドライブにセットする。この場合は、両方とも既にセット済みである。

※ 複写元のディスクと複写先のディスクを両ドライブにセットした後は、コピーの作業が完了するまで、途中でディスクの差し替えを行なう場面はない。 この作業中にディスクを差し替えると、ディスクが破壊される原因になるので、厳に慎むこと。

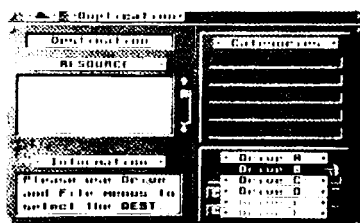
2) ドライブ指定パネルを開いて、複写元のディスクがセットされているドライブ名を選択する。この場合“A”を選択する。

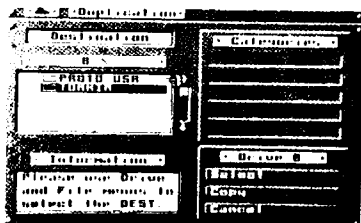
(キーボードの場合) [↑],[↓]キーで🖱をドライブ指定パネルに合わせて[↵]キーを押し、パネルを開く。更に[↑],[↓]キーで [Drive A:] を反転させ(下図)、[↵]キーを押す。

(マウスの場合) ドライブ指定パネルをクリックしてこれを開き、クリック・ボタンを押したまま🖱を動かして [Drive A:] が反転したところで(下図)、ボタンを離す。

3) ドライブを変更すると、下図のように変更先のドライブにセットされているディスク内のディレクトリがファイル・ウィンド内に表示されるので、この中からコピーするファイルの収まっているディレクトリを選択する。この場合“RESOURCE”を選ぶ。
(キーボードの場合) [←],[→]キーで☞をファイル・ウィンドに合わせ、[↑],[↓]キーで“RESOURCE”を反転させて[↵]キーを押す。
(マウスの場合) “RESOURCE”をクリックしてこれを反転させ、再度これをクリックする。

4) コピーするファイルのカテゴリーを選択する。ファイル・ウィンド内にファイルの一覧が表示されたら、目的のファイルを選択して、その頭にチェック・マークを付ける。この場合はまず [Tanks] を選択し、次に“TEAMTANK”を選ぶ。
(キーボードの場合) [←],[→],[↑],[↓]キーで☞を [Tanks] に合わせて[↵]キーを押し、次に☞をファイル・ウィンドに戻して [↑],[↓]キーで“TEAMTANK”を反転させてから[↵]キー。解除は再度[↵]キー。
(マウスの場合) [Tanks] をクリック。次に“TEAMTANK”をクリックしてこれを反転させ再度クリック。解除は更にクリック。

この際、同じカテゴリーのファイルであれば、チェック・マークを付けることによって、複数のファイルを1回の処理でコピーすることができる。

5) コピーするファイルの指定が済んだら、☞を Destination に合わせて⏎キーを押し、次にドライブ指定パネルを開いて、複写先のディスクがセットされているドライブ名を選択する。（下図参照）この場合は“B”を選択する。

（キーボードの場合）☞を Destination に合わせて⏎キーを押し、次にドライブ指定パネルを開いて☞を Drive B: に合わせ、⏎キーを押す。

（マウスの場合）まず Destination をクリックし、次にドライブ指定パネルをクリックしたまま☞を動かし Drive B: が反転したところでクリック・ボタンを離す。

※ 同一ディスク上の他のディレクトリにコピーする場合は、5）の Destination を選択し終えた時点で、ディレクトリ表示パネルを開いて一旦、ルート・ディレクトリに戻る。
（キーボードの場合）☞をディレクトリ表示パネルに合わせ⏎キー。次に ＊:＼ (＊:ドライブ名)を反転させ⏎キー。
（マウスの場合）ディレクトリ表示パネルをクリックして、＊:＼ (＊:ドライブ名)が反転したところでボタンを離す。
これ以降の操作は 6)以下に同じ。

6) この時点でファイル・ウィンドに複写先のディスク内のディレクトリが表示されるので、3)と同じ要領で複写先のディレクトリを選択する。この場合“ＴＯＮＫＩＮ”を選択する。

7) 最後に [Copy] を選択してコピーを実行する。
（キーボードの場合）☜を [Copy] に合わせて⏎キー。
（マウスの場合）[Copy] をクリック。

これでコピー完了である。
このモジュールを出る場合は、[Duplication] の [Exit Duplicator] を選択する。

> ※ ソース・ファイルのコピーに関してはプログラム・リストの機密保持という観点から、コピーを拒否される場合がある。
> OMEGA SYSTEM はソース・ファイルの一部にその設計者のパスワードを記録しており、ＩＤディスクにアクセスした際のパスワードと、このパスワードが一致した場合にに限り、ソース・ファイルのコピーが行なえるシステムになっている。

★ ９８ＮＯＴＥの場合（１ドライブ＋ＲＡＭドライブ）

このモジュールを使って２枚のＩＤディスク間でファイル・コピーを行なうためには、２台のドライブを必要とします。
９８ＮＯＴＥをご使用の方は、まずファイルの送り手側のディスクを“９８ＮＯＴＥメニュー”の“ＦＤ→ＲＡＭドライブコピー”でＲＡＭドライブにコピーしてからお使い下さい。以降の操作は２ドライブマシーンの場合と変わりません。
なお、同一ディスク上でのディレクトリ間のファイル・コピーは、１ドライブでも可能です。

************** サンプル・ファイルの利用について **************

“ＯＭＥＧＡ”のシステム・ディスク内には、サンプル・ファイルを収めた３つのディレクトリ “PROTO.USR” “RESOURCE” “TANK” が設けられています。ファイルの内容とその利用法は次の通りです。

“PROTO.USR”
ここのファイルは全てＩＤディスクの作成工程で、ＩＤディスクにファイル・コピーされたものです。

“RESOURCE”
ここには、等級評価の際に登場する敵車両や、チーム・コンバット用に設計された車両のソース・ファイル、及び実行ファイルなどが保管されています。
チーム・コンバットのサンプル・プログラムを見たい場合は、ファイル名 “TEAMTANK” “TH-GUARD” “TH-SOLD” のソース・ファイルをロードして下さい。
また、シミュレーション・デザイン・ファイルの “TH-TEAM” を戦闘シミュレーションにかけると、チーム・コンバットのデモを見ることができます。

“TANK”
ここには、等級“ＯＭＥＧＡ”(最上ランク)に達したエンジニアが作成したサイバータンクの実行ファイルが数十収められています。
等級“ＯＭＥＧＡ”に達した方はこれらの車両を相手に更なる研鑽を積み重ねて下さい。

以上で、“ＯＭＥＧＡ”の基本機能に関する解説は終了です。

PART 2

この章は、メニュー内の全項目の機能について、モジュールごとに記載する。

SECTION 1
メニュー項目の解説

1.1 External Control Module (ECM)

1.1.1 "≡" メニュー

About OMEGA　著作権、及び主システム設計者氏名の表示。

Analogue / L.C.D　ディスプレイ・モードをアナログ表示(16色)か液晶表示（L.C.D.）に設定する。

Keyboard　キーボードを選択する。

Mouse　マウスを選択する。

1.1.2 “Employee” メニュー

New Employee　セキュリティ·クリアランスに戻り、ＩＤディスクに再アクセスする。

Clearance Evaluation　等級評価を実行する。（p.163 参照）

Continue Evaluation　等級評価の途中データを記録したファイルを呼び出し、これを再開する。

Delete Saved Evaluation　等級評価の途中データを記録したファイルを削除する。

Call it a day　“ＯＭＥＧＡ”を終了する。

1.1.3 "Simulate" メニュー

Start a Simulation	Combat Simulation Module を用いて、戦闘シミュレーションを実行する。 (p.62 参照)
Continue a Simulation	Combat Simulation Module が実行した戦闘シミュレーションの途中データを記録したファイルを呼び出し、これを再開する。
Design a Simulation	シミュレーション・デザイン・ファイルを作成する。(p.58 参照)
Print Simulation Stats	Combat Simulation Module が実行した戦闘シミュレーションの各車両の戦績データをプリント・アウトする。 (p.68 参照)
Delete Simulation Design	シミュレーション・デザイン・ファイルを削除する。
Delete Saved Simulation	Combat Simulation Module が実行した戦闘シミュレーションの途中データを記録したファイルを削除する。

1.1.4 “Design” メニュー

Design Cybertank	Design Control Module にアクセスしてサイバータンクのソース・ファイルの開設、及び修正を行なう。
Test Cybertank	Cybertank Test Module を用いて、戦闘シミュレーションを実行する。（p.153 参照）
Design Battlefield	オリジナルの戦場ファイルを作成する。（p.70 参照）
Duplication Module	データ・ファイルを他のディレクトリにファイル・コピーする。（p.165 参照）
Delete Cybertank	サイバータンクの実行ファイルを削除する。

1．2　Design Control Module

1．2．1 “Cybertank” メニュー

New　サイバータンクのソース・ファイルを新しく開設する。（p.37 参照）

Load　サイバータンクの既存のソース・ファイルをロードする。（p.41 参照）

Save　ロード中のサイバータンクのソース・ファイルをＩＤディスクにセーブする。（p.41 参照）

Save as　ロード中のサイバータンクのソース・ファイルに新しい名前を付け、元のファイルとは別のソース・ファイルを開設する。（p.79 参照）

Chassis　Chassis Design Module にアクセスして、サイバータンクのシャシーを設計する。（p.38 参照）

AI　AI Module にアクセスして、サイバータンクのＡＩプログラムを作成する。（p.48, 92 参照）

Authorize　サイバータンクのソース・ファイルをマシン語に変換して実行ファイルを作成する。この際ソース・データに設計上の誤りや漏れがあれば、エラー・メッセージを出してこれを指摘する。（p.53 参照）

Print　ロード中のソース・ファイルのデータをプリント・アウトする。プリント・アウトされるものは、シャシーの仕様一覧、及びＡＩプログラムの全リストである。

Delete　　サイバータンクのソース・ファイルを削除する。
なお、削除するファイルが画面にロード中の場合は、Design Control Module を出る際、"Save changes to *** ?" に対して No を選択する。

※ 一旦削除したファイルは再生不能。

Exit　　External Control Module に戻る。

1．2．2 “Edit” メニュー

Cut　　反転させたテキストをＡＩプログラムから削除し、一時記憶領域に保存する。（p.83 参照）

Copy　　反転させたテキストを一時記憶領域に保存する。（p.83 参照）

Paste　　一時記憶領域にあるテキストをカーソルのある位置に挿入する。（p.84 参照）

Clear　　反転させたテキストを、一時記憶領域の内容を書き換えることなくＡＩプログラムから削除する。（p.84 参照）

Select All　　ＡＩプログラム上の全てのテキストを反転させる。

Expanded Text　　ＣＣＬ作成パネルの使用に際し、Expanded Text がオンの状態（頭にチェック・マークが付く）になっていると、コマンドは“拡張形”で表記され、オフの状態だと“標準形”で表記される。
初期設定は、オンになっている。（p.141 参照）

Undo　　直前の処理をキャンセルする。（p.84 参照）

1.2.3 "Capsule" メニュー

ＡＩカプセルの作成、及び使用に関する詳しい説明は PART 4 を参照のこと。

New	カプセル・ルーチンのソース・ファイルを開設する。(p.232 参照)
Load	カプセル・ルーチンのソース・ファイルをロードする。
Include	カプセル・ルーチンのプログラム・リストをＡＩボード上のカーソルのある位置に挿入する。(p.232 参照)
Save	カプセル・ルーチンのファイルをセーブする。
Verify	作成したカプセル・ルーチンのＡＩプログラムをマシン語に変換する。この際、ソース・データに設計上の誤りがある場合は、エラー・メッセージを出してこれを指摘する。
Delete	カプセル・ルーチンのファイルを削除する。
Print	ロード中のカプセル・ルーチンのプログラム・リストをプリント・アウトする。

1.3　Simulation Design Module

1.3.1 "Design" メニュー

Select Cybertank	チーム・コンバットのシミュレーション・デザイン画面から通常のシミュレーション・デザイン画面に戻る。
Select Teams	チーム・コンバットのシミュレーション・デザイン画面に入る。（p.244 参照）
Position Headquarters	チーム・コンバットのシミュレーション・デザイン・ファイル作成の過程で、マップ上に司令部（ＨＱ）を配置する。（p.245 参照）
Load Simulation Design	シミュレーション・デザイン・ファイルをロードする。
Save Simulation Design	ロード中のシミュレーション・デザイン・ファイルをＩＤディスクにセーブする。（p.61 参照）
Exit Setup	External Control Module に戻る。

1.4 Combat Simulation Module

1.4.1 "Simulation" メニュー

Restart	ロード中の戦闘シミュレーションをパラメータ類を初期値に戻して、再スタートさせる。 また Position Cybertank 機能でタンクの配置を変更した後、戦闘を再開させるときに選択する。(P.67 参照)
Set Number of Battles	戦闘シミュレーションの実行回数を設定する。(p.64 参照)
Position Cybertank	戦闘シミュレーションでサイバータンクの位置や車両の向きを手動で任意に設定する。(p.66 参照)
Satellite View	サテライト・モードに切り替え、戦場全域をモニターする。サテライト・モードのキャンセルは再度これを選択する。
Display Numbers	シミュレーション・ウィンド内の車両をその"認識数字"で表示する。 "認識番号"に関しては p.64 参照。
Display Graphics	これをオフにすると、シミュレーション・ウィンド内のディスプレイが消えて、戦闘シミュレーションの処理速度が上がり、戦闘時間を短縮することができる。

Sound On	これをオフにすると、端末のスピーカーがオフになり、シミュレーションの処理速度が上がる。
Save Simulation	戦闘シミュレーション処理中の途中データを、新しく名前を付けて作成したファイルにセーブする。
Exit Simulation	External Control Module に戻る。

1.5　Cybertank Test Module

1.5.1 “Debugger” メニュー

Restart	ロード中の戦闘シミュレーションをパラメータ類を初期値に戻して再スタートさせる。
Satellite View	サテライト・モードに切り替え、戦場全域をモニターする。
Sound On	これをオフにすると、端末のスピーカーがオフになり、戦闘シミュレーションの処理速度が上がる。
Trace Mode On	これをオンにすると Primary Tank の動きを処理しているＡＩプログラムが画面の右にスクロール表示される。（p.155 参照）
Single Step On	これをオンにすると戦闘シミュレーションの処理にポーズがかかり、スペースキーを１回押すごとに、１コマンドずつ処理が進行する。（p.156 参照）
Exit	External Conrtol Module に戻る。

1.6 Clearance Evaluation Module

1.6.1 "Evaluation" メニュー

Satellite View	サテライト・モードに切り替え、戦場全域をモニターする。
Display Graphics	これをオフにすると、シミュレーション・ウィンドのディスプレイが消えて、戦闘シミュレーションの処理速度が上がり、戦闘時間を短縮することができる。
Sound On	これをオフにすると、ターミナルのスピーカーがオフになって処理速度が上がり、評価時間を短縮することができる。
Save Evaluation	等級評価処理中の途中データを、新しく名前を付けて作成したファイルにセーブする。
Exit Evaluation	External Control Module に戻る。

1.7　Design Battlefield Module

1.7.1 "Battlefield" メニュー

Load	戦場ファイルをロードする。（p.73 参照）
Save	ロード中の戦場データを、ＩＤディスクにセーブする。（p.73 参照）
Delete	戦場ファイルを削除する。
Print	戦場ファイルをプリント・アウトする。
Satellite View	サテライト・モードに切り替え、戦場全域をモニターする。
Fill Map	指定したタイルで戦場全体を埋める。（p.72 参照）
Fill Screen	指定したタイルでエディット・ウィンド内に表示さたエリアを埋める。（p.72 参照）
Clear Map	戦場全体を、なにも入力されていない初期の状態に戻す。（p.72 参照）
Exit	External Control Module に戻る。

1.7.2 "Block" メニュー

Load　ブロックのファイルをロードする。操作は戦場ファイルのロードに準じる。(p.73 参照)

Save　作成したブロックを、IDディスクにセーブする。操作は戦場ファイルのセーブに準じる。(p.73 参照)

Delete　ブロックのファイルを削除する。

Clear Block　パーツ表示エリア内のブロック表示を消す。(p.75 参照)

Clear Copy　戦場マップの一部を、エディット・ウィンドド上からパーツ表示エリアにコピーするために行なった範囲指定をキャンセルする。(p.75 参照)

1.7.3 "Edit" メニュー

Display Tiles　パーツ表示エリアにタイルを表示する。

Display Blocks　パーツ表示エリアにブロックを表示する。

Pen Down　鉛筆書き機能：指定したタイルをエディット・ウィンドド中央に反転表示し、戦場マップを移動させることによってそのタイルを連続的に配置する。（p.72 参照）

Plop mode on　指定したタイルやブロックを戦場マップに書き込むとき選択する。（p.71,76 参照）

Copy mode on　ブロック作成の過程で、エディット・ウィンドド上からパーツ表示エリアにコピーする部分の範囲指定を行なう。（p.75 参照）

1.8　Data Duplication Module

1.8.1 "Duplication" メニュー

Exit Duplicator　External Control Module に戻る。

PART 3

この章は、CCL コマンドの全てのヴァリエーションについて記載する。

SECTION 1
ＣＣＬコマンド総覧

SECTION BRIEF

ここでは、OMEGA SYSTEM が処理することのできる全種類の CCL コマンドについて記載する。

尚、以下はこのセクションのコマンド表記における略号の解説である。

★ コマンド表記の規約

cyc	コマンドのサイクル数を示す。
[ ]	この括弧内の単語は、省略可能であることを示す。括弧内の単語を省略せずに表記したものが Expanded Text である。(p.141 参照)
(not)	これが挿入されているコマンドは、肯定形と否定形の２種類があることを表わす。
#	システム·ヴァリアブル、ユーザー·ヴァリアブル、または数値のいずれかを表わす。
"Label"	ラベル名を表わす。
X	Battlefield の x座標の値を表わす。
Y	Battlefield の y座標の値を表わす。
<seq.com>	"branch to Label" または "Do Label" を表わす。(p.221 参照)

1.1　サイバータンクの移動

1.1.1 トレッド・ダメージとその修理

● コマンド構造

```
 1cyc      If [tank] treads [are] (not) functional
                then <seq.com>
           (もし、トレッドが機能しているならば)
60cyc      Repair treads
           (トレッドを修復せよ)
```

● 接触するシステム・ヴァリアブル

TreadDamage

● 使用例

```
CheckTread
      If tank treads are functional then Branch to Ct_Exit
      Repair treads
Ct_Exit
      Resume
```

サイバータンクは移動や方向転換にトレッド(キャタピラ)を使用する。トレッドが破壊されれば、サイバータンクは移動できない。もし購入できれば、修理用キットを使うと、いつでも損傷したトレッドを修理することができる。サイバータンクの修理に関しては、p.211 を参照。

1.1.2 移動

● コマンド構造

```
40cyc     Move  [tank]  forward   #
40cyc     Move  [tank]  backward   #
```

● 接触するシステム・ヴァリアブル

TankX
TankY

● 使用例

```
Move_Clear
      Detect obstruction at tank direction
      If Movement is not obstructed then Branch to MC_Move
      Fire weapon at obstruction
      Branch to Move_Clear
MC_Move
      Move tank forward 1
      Resume
```

サイバータンクは、前方（車両の向いている方角）又は、後方に移動することができる。そこで、あなたはタンクの移動距離を指定しなければならない。移動できる範囲は、０～６３(hm)までである。
一旦移動が指令されると、サイバータンクは目的地へ着くか、障害物にぶつかるまで止まらない。障害物に衝突すると、サイバータンクは損傷をうけて止まる。損傷の程度はぶつかった物体によって決まる。サイバータンクが移動すると燃料を消費する。

1.1.3 方向転換

● コマンド構造

```
16cyc    Turn  [tank]  left  #
16cyc    Turn  [tank]  right  #
16cyc    Turn  [tank]  to  "angle"
16cyc    Turn  [tank]  to  X  Y
16cyc    Turn  [tank]  to  face  [enemy] tank
16cyc    Turn  [tank]  to  face  enemy HQ
16cyc    Align  tank  [with scanner]
```

● 接触するシステム・ヴァリアブル

TankDir

● 使用例

```
Find_Open
       Detect obstruction at tank direction
       If movement is not obstructed then branch to Fo_Move
       Turn tank right 1
       Branch to find_Open

Fo_Move
       Move tank forward 1
       Resume
```

サイバータンクは絶対方位の8方向全てに向けて方向転換ができる。指定された座標点や目標物が、絶対方位の間に位置する場合は、その方角に最も近い絶対方位を算出して、そちらに方向転換する。

1.1.4 障害物の探査

● コマンド構造

```
4cyc    Detect  [obstruction]  at  #
4cyc    Detect  [obstruction]  at  tank  direction
4cyc    Detect  [obstruction]  at  scanner  direction
1cyc    If [movement is] (not) obstructed  then <seq.com>
1cyc    If  obstruction [is]  enemy HQ  then <seq.com>
1cyc    If  obstruction [is]  ally HQ  then <seq.com>
```

● 接触するシステム・ヴァリアブル

```
ObstacleX            ObstacleDist
ObstacleY            ObstacleType
```

● 使用例

```
Move_Clear
      Detect obstruction at tank direction
      If movement is not obstructed then branch to MC_Move
      Fire weapon at obstruction
      Branch to Move_Clear
MC_Move
      Move tank forward 1
      Resume
```

サイバータンクには全て移動センサーが装備されており、これはある特定の方角に障害物があるかどうかを探査する。障害物とは、衝突によりサイバータンクに損傷を与えうるあらゆる物体のことである。進路上の障害物を砲撃によって破壊しながら移動する方法は、障害物を避けて移動する方法に比べ通常早い。

1.1.5 車両の向きを判断する

● コマンド構造

```
1cyc    If  tank [is] (not) facing [enemy] tank
            then <seq.com>
1cyc    If  tank [is] (not) aligned [with scanner]
            then <seq.com>
1cyc    If  tank [is] (not) facing enemy HQ  then <seq.com>
1cyc    If  tank [is] (not) facing  X Y  then <seq.com>
```

● 接触するシステム・ヴァリアブル

無し

● 使用例

```
Move_Clear
     Detect obstruction at scanner direction
     If movement is not obstructed then branch to MC_ChkDir
     Fire weapon at obstruction
     Branch to Move_Clear
MC_ChkDir
     If tank is aligned with scanner then branch to MC_Move
     Align tank with scanner
MC_Move
     Move tank forward 1
     Resume
```

使用例は、タンクをスキャナーの向いている方角に移動させようとするルーチンである。７行目のスキャナーとの同列化をチェックするコマンドは、車両がたまたまスキャナーと同列であった場合、無条件に“Align tank with scanner”を処理することによって、無駄なサイクルを浪費することを防止するためのものである。

1.2　スキャナーの使用

スキャナーの最も重要な機能は、迅速にエリアを探査し、その結果をサイバータンクのメイン・コンピュータに返すことにある。
スキャナーは、敵車両、司令部、各種の地形構成を探知する能力を持ち、サイバータンクの設計に重要な役割を果たす。
なお、Shield を上げると、全スキャナーの探査域は半減する。

1.2.1 スキャナー・ダメージとその修理

● コマンド構造

1cyc　　If scanner [is] (not) functional then <seq.com>
　　　　（もし、スキャナーが機能しているならば）

60cyc　　Repair scanner
　　　　（スキャナーを修理せよ）

● 接触するシステム・ヴァリアブル

ScanDamage

● 使用例

```
Check_Scan
      If scanner is functional then branch to CS_Exit
      Repair scanner
CS_Exit
      Resume
```

1.2.2 敵車両の探査

● コマンド構造

```
8cyc    Scan for [enemy] tank
1cyc    If [enemy] tank [was] (not) found then <seq.com>
```

● 接触するシステム・ヴァリアブル

```
EnemyX        EnemyDist
EnemyY
```

● 使用例

```
Find_Tank
     Rotate scanner right 1
     Scan for enemy tank
     If enemy tank was not found then  branch to Find_Tank

FT_FoundIt
     Resume
```

1.2.3 物体の探査

● コマンド構造

```
8cyc    Scan  for [closest] object
1cyc    If [closest] object [was](not) found then <seq.com>
```

（"closest object" の定義に関しては p.95 参照）

● 接触するシステム・ヴァリアブル

ObjX　　ObjY　　ObjType　　ObjDist

● 使用例

```
Clear_Area
   Tree = 4
   House = 5
   Align tank with scanner
CA_Loop
   Scan for closest object
   If closest object was not found then branch to CA_Rotate
   If ObjType = Tree then CA_Destroy
   If ObjType = House then CA_Destroy
CA_Rotate
   Rotate scanner right 1
   If tank is not aligned with scanner then branch to
       CA_Loop
CA_Exit
   Resume
CA_Destroy
   If closest object is beyond weapon range then branch to
       CA_Rotate
   Fire weapon at closest object
   Branch to CA_Loop
```

1.2.4 指令部の探査

● コマンド構造

```
8cyc      Scan  for  enemy HQ
1cyc      If [closest] object [is] enemy HQ  then <seq.com>
1cyc      If [closest] object [is] ally HQ  then <seq.com>
1cyc      If  enemy HQ [was] (not) found  then <seq.com>
```

● 接触するシステム・ヴァリアブル

```
EnemyHQX           EnemyHQDist
EnemyHQY
```

● 使用例

```
Check_Base
      Scan for enemy HQ
      If enemy HQ was found then ShootIt
      Rotate scanner  1
      Branch to Check_Base
ShootIt
      Fire weapon at enemy HQ
      Resume
```

指令部を配置してチーム・コンバットを行なう場合、味方の指令部の座標位置は初めからチーム員のシステム・ヴァリアブルにセットされている。

1.2.5 スキャナーの旋回

● コマンド構造

```
16cyc    Rotate  [scanner]  left  #
16cyc    Rotate  [scanner]  right  #
16cyc    Rotate  [scanner]  to  "angle"
16cyc    Rotate  [scanner]  to  X  Y
16cyc    Rotate  [scanner]  to face  [enemy] tank
16cyc    Align  scanner [with tank]
 1cyc    If  scanner [is] (not) aligned [with tank]
              then <seq.com>
```

● 接触するシステム・ヴァリアブル

ScanDir

● 使用例

```
Find_Tank
        Rotate scanner right 1
        Scan for enemy tank
        If enemy tank was found then branch to FT_FoundIt
        Branch to Find_Tank
FT_FoundIt
        Resume
```

1.2.6 スキャナー・ロック

● コマンド構造

```
5cyc    Lock [scanner]   （スキャナー・ロックせよ）
5cyc    Unlock [scanner] （スキャナー・ロックを解除せよ）
1cyc    If [scanner is] locked then <seq.com>
        （もし、敵をスキャナー・ロックしているならば）
1cyc    If [scanner is] unlocked then <seq.com>
        （もし、スキャナー・ロックが解除されたならば）
```

● 接触するシステム・ヴァリアブル

無し

● 使用例

```
Find_Tank
      Scan for enemy tank
      If enemy tank was found then FT_Found
      Rotate scanner right 1
      Branch to Find_Tank
FT_Found
      Lock scannner
      If scanner is unlocked then branch to Find_Tank
      Resume
```

スキャナー・ロックは Special Items の "Scanner Lock" を装備しているとき作動するもので、最後に探知した目標物をスキャナーが自動的に追尾し、捕捉し続ける機能である。但し、対象が探査域外に出てしまった場合や、スキャナーと対象との間にタイプ・ナンバー4以上の物体が入った場合は解除される。任意解除するときは "Unlock" を使用する。

1.2.7 敵にスキャナー・ロックされているか否かをチェックする

● コマンド構造

```
lcyc    If  [tank is] (not) being scanned  then <seq.com>
        (もし、敵にスキャナー・ロックされていたら)
```

● 接触するシステム・ヴァリアブル

無し

● 使用例

```
Detect
      If tank is not being scanned then branch to DT_No
      Jam scanner signal
DT_No
      Resume
```

敵があなたのサイバータンクをスキャナー・ロックしているか否かを識別するためには、Special Items の "Listener" を装備していなければならない。
上のルーチンは Listener と Jammer の両方を装備していることを前提に組まれている。"Jam" コマンドは次頁で解説。

1.2.8 敵のスキャナー・ロックを妨害電波で解除する

● コマンド構造

```
8cyc    Jam  [scanner signal]
        (敵のスキャナー・ロックを解除せよ)
```

● 接触するシステム・ヴァリアブル

無し

● 使用例

```
Detect
      If tank is not being scanned then branch to DT_No
      Jam scanner signal
DT_No
      Resume
```

"Jam" コマンドは、サイバータンクが Special Items の "Jammer" を装備している場合にのみ機能する。但し、Listener の装備は必要としない。

1台のサイバータンクがこのコマンドを使用すると、自分を含め、敵味方全ての車両のスキャナー・ロックが解除される。

1.2.9 リモート・スキャナーの使用

● コマンド構造

```
15cyc    Launch [remote scanner]
         (リモート・スキャナーを発射せよ)
 1cyc    If remote [scanner is] available then <seq.com>
         (リモート・スキャナーが使用できるならば)
 1cyc    If remote [scanner is] unavailable then <seq.com>
         (リモート・スキャナーが使用できないならば)
```

● 接触するシステム・ヴァリアブル

```
EnemyX          EnemyY
EnemyDist       RemotesLeft
```

● 使用例

```
Try_Launch
      If remote scanner is unavailable then branch to TL_No
      Launch remote scanner
      Turn tank to EnemyX EnemyY
TL_No
      Resume
```

・リモート・スキャナーは Special Items の "Launcher" を装備している場合にのみ使用できる。
・ Launcher 1基に4発リモート・スキャナー装填されている。
・リモート・スキャナーは発射されると、最も近くにいる敵車両の位置を捕捉し、そのデータを OSIの静止衛星を経由してあなたのタンクのメイン・コンピュータにセットする。
・リモート・スキャナーは指令部の探知に使用することはできない。

1.3 武器の使用

武器はその種類によって発射速度と破壊力が異なる。
なお、Shield が上がっているときは、いかなるサイバータンクも武器を使用することができない。

1.3.1 武器のダメージとその修理

● コマンド構造

```
 1cyc      If weapon [is] (not) functional then <seq.com>
           (もし、武器が機能しているならば)
60cyc      Repair weapon
           (武器を修理せよ)
```

● 接触するシステム・ヴァリアブル

WeapDamage

● 使用例

```
Check_Weap
      If weapon is functional then branch to CW_Exit
      Repair weapon
CW_Exit
      Resume
```

1.3.2 目標物が射程内にあるか否かを判断する

● コマンド構造

```
lcyc    If  [enemy] tank [is] within [weapon] range
            then <seq.com>
lcyc    If  [enemy] tank [is] beyond [weapon] range
            then <seq.com>
lcyc    If  [closest] object [is] within [weapon] range
            then <seq.com>
lcyc    If  [closest] object [is] beyond [weapon] range
            then <seq.com>
lcyc    If  enemy HQ [is] within [weapon] range
            then <seq.com>
lcyc    If  enemy HQ [is] beyond [weapon] range
            then <seq.com>
```

● 接触するシステム・ヴァリアブル

無し

● 使用例

```
Find_Tank
      Scan for enemy tank
      If enemy tank was found then branch to Check_Range
      Rotate scanner right 1
      Branch to Find_Tank
Check_Range
      If enemy tank is beyond weapon range then TooFar
      Fire weapon at enemy tank
      Branch to Find_Tank
TooFar
      Resume
```

1.3.3 武器の発射

● コマンド構造

```
40cyc     Fire  [weapon]  at  [enemy] tank
40cyc     Fire  [weapon]  at  [closest] object
40cyc     Fire  [weapon]  at  obstruction
40cyc     Fire  [weapon]  at  X  Y
40cyc     Fire  [weapon]  at  tank direction
40cyc     Fire  [weapon]  at  scanner direction
40cyc     Fire  [weapon]  at  enemy HQ
```

● 接触するシステム・ヴァリアブル

無し

● 使用例

```
Clear_Path
       Detect obstruction at tank direction
       If movement is not obstructed then LetsMove
       Fire weapon at obstruction
LetsMove
       Move tank forward 1
       Resume
```

武器を目標物に対してではなく、特定の方角に向けて使用した場合、弾頭はその進路上の最初の障害物に着弾する。進路上に障害物がない場合、弾頭は射程域の限界点に着弾する。

1.4　その他のコマンド

ここでは特殊アイテムの使い方、キー入力による手動操作、デバッグ用のコマンドなど、これまでに触れられなかったコマンドついて記載する。

1.4.1 ダメージの修理

● コマンド構造

60cyc	Repair　Internal （電気系統を修理せよ）
60cyc	Repair　Armor （装甲を修理せよ）
60cyc	Repair　Treads （トレッドを修理せよ）
60cyc	Repair　Scanner （スキャナーを修理せよ）
60cyc	Repair　Weapon （武器を修理せよ）
1cyc	If　[Repair] Kit [is] available　then <seq.com> （もし、修理用キットが使用できるならば）
1cyc	If　[Repair] Kit [is] unavailable　then <seq.com> （もし、修理用キットが使用できないならば）

● 接触するシステム・ヴァリアブル

IntDamage	TreadDamage
ArmorDamage	WeapDamage
ScanDamage	KitsLeft

● 使用例

```
TreadFixer
      If tank treads are not functional then FixTreads
      Resume
FixTreads
      If Repair Kit is available then FixIt
      Resume
FixIt
      Repair Treads
      Resume
```

Special Items の“Repair Kit”を装備していると、損傷したコンポーネントを修理することができる。但し、その修復能力には一定の限界がある。
修理用キットは４回使用することができ、それぞれが任意のコンポーネントの修理に充てることができる。

1.4.2 シールド

● コマンド構造

```
10cyc     Raise [shield]  (シールドを上げろ)
 5cyc     Lower [shield]  (シールドを下げろ)
 1cyc     If  shield [is] up  then <seq.com>
          (もし、シールドが上がっているならば)
 1cyc     If  shield [is] down  then <seq.com>
          (もし、シールドが下がっているならば)
```

● 接触するシステム・ヴァリアブル

無し

● 使用例

```
Chk_Shield
       If shield is down then ShootIt
       Lower shield
SootIt
       Fire weapon at enemy tank
       Resume
```

シールドは、Special Items の "Shield" を装備している場合にのみ使用できるもので、これを上げていると、敵から攻撃を受けたときのダメージを最小限に抑えることができる。但し、シールドを上げると次のような不利益が発生する。

★ スキャナーの探査域が半減する。
★ 武器が使用できない。
★ 消費燃料が増大する。

1.4.3 万策尽きたとき

● コマンド構造

```
lcyc      Self  Destruct     (自爆せよ)
```

● 接触するシステム・ヴァリアブル

全て

● 使用例

```
CheckAll
        If tank treads are functional then branch to OK
        If scanner is functional then branch to OK
        If weapon is functional then branch to  OK
        If Repair Kit is available then branch to OK
        Self Destruct
OK
        Resume
```

このコマンドはサイバータンクの機能が万事休した場合、残された唯一の高潔な行為として自爆を決行する。

1.4.4 ある座標点までの距離を測定する

● コマンド構造

```
1cyc    Get  Distance [to]  X  Y
        (座標点 x,y までの距離を測定せよ)
```

● 接触するシステム・ヴァリアブル

XYDist

● 使用例

```
CheckDist
        Get Distance to 62 62
        If XYDist > 30 then MaybeGo
        Branch to CheckTank
```

このコマンドは、Battlefield 上のある特定の座標点までの距離を測定し、この値をその後の行動の判断材料とするときに用いる。

1.4.5 ランダム値をとる

● コマンド構造

```
1cyc    Get   Random          ( 1～100 の間でランダム値をとれ)
1cyc    Get   Random  to  #   ( 1～ #  の間でランダム値をとれ)
```

● 接触するシステム・ヴァリアブル

RandomNum

● 使用例

```
Maybe_Go
        Get Random
        If RandomNum > 50 then Turn_Left
        Turn tank right 1
        Branch to Move_It
```

このコマンドは、次の行動の判断をランダムな要素に委ねるときに使用するもので、使用例はランダム値が５０より大きければ左に、５０以下であれば右に車両を旋回させるルーチンである。
なお、"#" の値域は 1～100 である。

1.4.6 ビープ音を鳴らす

● コマンド構造

1cyc　Beep　(ビープ音を鳴らせ)

● 接触するシステム・ヴァリアブル

無し

● 使用例

```
Found
      Lock scanner
      Beep
      If enemy tank is within weapon range then do Shoot
      Do Chase
      Branch to ScanTank
```

このコマンドはデバッグ用のコマンドで、あるコマンドが確かに処理されているか否かを判断するときのシグナルなどに用いられる。

1.4.7 コメント・ラインの挿入

● コマンド構造

```
0cyc    *          (この行を無視せよ)
```

● 接触するシステム・ヴァリアブル

無し

● 使用例

```
CS_Move1
* Move the tank closer to center
         Turn tank to 31 31
         Detect obstruction at tank direction
         If movement is obstructed then CS_Blocked
CS_Move2
         Move tank forward 1
         Branch to CS_Scan
```

“*” (アスタリスク)は、OMEGA SYSTEM に何らかの処理を行なわせる通常のコマンドとは異なる。
OMEGA SYSTEM は行の初めにこれを見つけると、その行をコメント・ラインと認識してこれを飛ばし、次の行の処理に移る。
この “*” の機能を利用して、作成したプログラムの適所に注釈を入れておけば、時間を経てからプログラムを見返したときの理解に役立つであろう。

1.4.8 ブレイク・ポイントの設定

● コマンド構造

```
Ocyc     Break     (処理を中断せよ)
```

● 接触するシステム・ヴァリアブル

無し

● 使用例

```
Check_Enemy
          Scan for enemy tank
          If enemy tank was not found then Done
          If enemy tank is beyond weapon range then Done
          Break
Done
          Resume
```

“Break” コマンドはデバッグ用コマンドで、Cybertank Test Module でのみ機能する。OMEGA SYSTEM はこのコマンドに会うと、そこで処理を中断して、Single Step On (p.156 参照)に切り替える。
なお、Combat Simulation Module ではこのコマンドは無視される。

1.4.9 手動制御

● コマンド構造

```
lcyc    If  [last] key [pressed]  then <seq.com>
        ( A~Z の内いずれかのキーが押されたら)
lcyc    If  [last] Key [pressed] = A~Z  then <seq.com>
        ( A~Z のうち、ある特定のキーが押されたら)
```

● 接触するシステム・ヴァリアブル

無し

● 使用例

```
Read_Key
      If  last key pressed = M  then branch to Move_Forwd
      If  last key pressed = F  then branch to Fire
      Resume
Move_Forwd
       Move tank forward 1
       Branch to Done
Fire
       Fire weapon at closest object
Done
       Resume
```

このコマンドはシミュレーション実行中、キーボードからのキー入力によって、サイバータンクを手動操作するものである。
なお、このコマンドが認識するキーはアルファベットのＡ～Ｚである。

1.4.10 シーケンス・コマンド

● コマンド構造

```
1cyc    Branch  to  "Label"
1cyc    Goto  "Label"          (= Branch  to  "Label" )
1cyc    Do  "Label"
1cyc    Gosub  "Label"         (= Do  "Label" )
1cyc    Resume
```

● 接触するシステム・ヴァリアブル

無し

● 使用例

```
TankProg
          Do ScanEnemy
          Do MoveFrwd
          Branch to TankProg
ScanEnemy
          Scan for enemy tank
          If enemy tank was found then do ShootIt
          Resume
MoveFrwd
          Move tank forward 1
          Resume
ShootIt
          If enemy tank is within weapon range
              then do KillIt
          Resume
KillIt
          Fire weapon at enemy tank
          Resume
```

1.4.11 カプセル・ルーチンを読み込む

● コマンド構造

```
Ocyc      Include "カプセル・ルーチンのファイル名"
```

● 接触するシステム・ヴァリアブル

無し

● 使用例

```
TankProg
        Do ScanEnemy
        Do MoveFrwd
        Branch to TankProg

        Include ScanEnemy
        Include MoveFrwd
```

※ "Include" コマンドの使用法については、PART 4 の "カプセル・ルーチンの使用法" を参照のこと。

1.4.12 通信リンクの使用

● コマンド構造

```
1cyc    Switch  [CommLink]  on
1cyc    Switch  [CommLink]  off
1cyc    Transmit  [Code]  #  [to team]
1cyc    Copy  [CommLink]  Data
1cyc    Clear  [CommLink]  Data
```

● 接触するシステム・ヴァリアブル

AllyNum	CopyNum
AllyCode	CopyCode
AllyX	CopyX
AllyY	CopyY
AllyDist	CopyDist
AllyDir	CopyDir
AllyEnemyX	CopyEnemyX
AllyEnemyY	CopyEnemyY
AllyEnemyDist	CopyEnemyDist
AllyEnemyDir	CopyEnemyDir

※ 通信リンク（Comm-Link）の使用に関する詳しい解説は、PART 5 の“チーム・コンバット”を参照のこと。

PART 4

この章は、OSI カプセル・ルーチンの使い方について記載する。

SECTION 1

カプセル・ルーチンの使用法

SECTION BRIEF

この節では、カプセル・ルーチンの使い方、及びその作成方法について述べる。

1.1　カプセル・ルーチンとは

“カプセル・ルーチン”とは、ある特定の任務を処理するために設計されたサブ・ルーチン（p.114 参照）であり、OMEGA SYSTEM には、このファイルを専門に保管する“カプセル・ライブラリ”と呼ばれる場所が設けられている。
エンジニアは作成したカプセル・ルーチンをここに置いて、いつでも必要なときにこれを呼び出し、作成中のＡＩプログラムに組み込むことができる。
カプセル・ルーチンはその適切な利用によって、手軽に高度なＡＩを作成することのできる、いわばＡＩプログラムのコンポーネントと言えよう。
なお、諸君のカプセル・ライブラリには、OSI 作成のカプセル・ルーチンが２１個、既に登録されている。

1.2　カプセル・ルーチンの使用法

カプセル・ルーチンの使用は“ＡＬＰＨＡ”の設計で既に経験したことだが、ここで改めて解説しておこう。

以下は、ファイル名を“Scrap”という攻撃用カプセル・ルーチンの全リストである。

★“Scrap”のプログラム・リスト★

```
Lock
        Scan for enemy tank
        If enemy tank was not found then  Lost
        If enemy tank is beyond weapon range then Face
        Lock scanner
        Fire weapon at enemy tank
        Branch to Lock
Face
        Turn tank to face enemy tank
        Do Watch_out
        Branch to Lock
Lost
        Resume

Watch_out
        Detect obstruction at tank direction
        If movement is not obstructed then Go_ahead
        Fire weapon at obstruction
        Detect obstruction at tank direction
        If movement is not obstructed then Go_ahead
        Turn tank left 1
        Branch to Watch_out
Go_ahead
        Move tank forward 1
        Resume
```

“Scrap” は､別の探査用ルーチンが敵を探知した時点でアクセスされることを前提に設計されたもので、その具体的任務は、敵が射程外であればこれを追跡し、射程内に入れ次第スキャナーをロックして砲撃。これを繰り返し、途中で敵を見失ってしまうか、敵が破壊された時点でメイン・ルーチンに “Resume” する、という内容のものである。
これをＡＩプログラムに組み込むためには、次のような作業を経なければならない。

1) まず､カプセル・ルーチンはサブ・ルーチンとして設計されたものなので、通常 “Do” コマンドによってアクセスされる。
一方、“Do” コマンドの後には必ずラベル名が来るので、 “Scrap” を実行させるためのコマンドは “Do” の後に “Scrap” ルーチンの最初のラベル名を入れて

```
Do  Lock
```

となる。

2) 一方、OMEGA SYSTEM はこのコマンドに会うとリスト上を駆け回ってラベル “Lock” を捜しに行く。 従って、本来ならばＡＩプログラムに “Scrap” のリストを書き込まなければならないが、これをたった１行で代行するのが “Include” コマンドである。
但し “Include” の後にはルーチンのファイル名を入れる決まりになっているので、このコマンドは

```
Include  Scrap
```

になる。

以上をまとめると、カプセル・ルーチンをＡＩプログラムに組み込むためには、次の２つのコマンドが必要であるということになる。

```
  Do      "カプセル・ルーチンの最初にアクセスされるラベル名"
Include   "カプセル・ルーチンのファイル名"
```

さて、次の問題はこれらのコマンドを本体のプログラムのどこに組み込めばよいかということである。
"Do Lock" をどこに入れるかはＡＩの仕様次第である。しかし、これに対応する "Include Scrap" は、入れてよい場所が限られる。以下は "Scrap" ルーチンを利用したＡＩプログラムの１例である。

```
      Begin
             Do Smell_out
             Do Lock
             Branch to Begin
①==>
      Smell_out
             Scan for enemy tank
             If enemy tank was found then branch to Done
             Rotate scanner left 1
             If scanner is aligned with tank then do Watch_out
             Branch to Smell_out
②==>
      Done
             Resume
③==>
```

お気付きの通り、このＡＩには "Include Scrap" が抜けている。

結論から先に言えば、このコマンドを書き込める場所は、リスト中の①、②、③と番号を付したところに限られる。と言うのは――
“Include Scrap”という１行が挿入されることは、OMEGA SYSTEM にとってコマンドの処理上、その場所に２５行の“Scrap”ルーチンのリストが存在するのと全く同じことを意味する。
いま仮に、左のＡＩの３行目“Do Lock”の次に“Include Scrap”を挿入したとすると、OMEGA SYSTEM はコマンド“Do Lock”の指示通り“Scrap”ルーチンを処理し，“Resume”でメイン・ルーチンに再帰したあと、今度はコマンド“Include Scrap”の処理にとりかかる。
ところがこの１行は、２５行の“Scrap”のリストに他ならないので、OMEGA SYSTEM はここで再び“Scrap”ルーチンを処理することになり処理が重複してしまう。
従って“Include Scrap”は、いかなる場合においても OMEGA SYSTEM が直接アクセスすることのない行に置かなければならない。それが即ち、①、②、③の場所なのである。

1.3　カプセル・ルーチンの作り方

1) メニュー Design の Design Cybertank を選択する。
2) メニュー Capsule の New を選択する。
3) ルーチンの名前を入力し、Save を選択する。
4) ＡＩボード上でルーチンを作成する。
   カプセル・ルーチンはサブ・ルーチンとして使用されるので、必ず"Resume"で終了する構造にする。
5) 作成を終えたら、メニュー Capsule の Verify を選択する。
   "Verify"は、ＡＩプログラムの"Authorize"に相当する機能で、ルーチンに誤りがあればこれを指摘してくれる。
6) "VERIFICATION COMPLETED"の表示が出たら完了である。
7) ECM を選択し、"Save changes to *** ?"の表示が出たら Yes を選択してソース・リストをセーブする。

1.4　カプセル・ルーチンのリストの書き込み方

カプセル・ルーチンを、"Include"コマンドによって組み込むのではなく、そのリストを直接プログラムに書き込む場合は、以下の手順を踏む。

1) まずカプセル・ルーチンを組み込む本体のＡＩプログラムをロードする。
2) カプセル・ルーチンのリストを挿入する箇所にカーソルを置く。
3) メニュー Capsule の Include を選択して、書き込むカプセルのファイルを選ぶ。

SECTION 2

ＯＳＩカプセル・ルーチンの使用法

SECTION BRIEF

この節では OMEGA SYSTEM が諸君のＩＤディスクにあらかじめコピーしておいた OSI カプセル・ルーチンの使い方について解説する。

２．１　カプセル・ヴァリアブル

OSI カプセル・ルーチンは、これをＡＩの一部に取り入れたり、或いはこれらを組み合わせることによって、簡単にＡＩが作成できることを目的に開発されたサブ・ルーチンである。
OSI カプセル・ルーチンには、共通の定義で使用されている７種類のユーザー･ヴァリアブルがあり、これを“カプセル･ヴァリアブル”と呼ぶ。
カプセル･ヴァリアブルは、いわば既製品であるところの OSI カプセル・ルーチンをその時々の用途に適応させたり、アルゴリズムを決定する際に使用する重要な道具であり、OSI カプセル・ルーチンを効果的に使用するためにはこの使い方を理解していなければならない。

カプセル・ヴァリアブルは、その機能によって２種類に分類することができる。１つは、カプセル・ルーチンを呼び出す際に“ユーザー”が前もって値を設定してやらなければならないもの。もう１つは、カプセル・ルーチンがいかなる状況の下で同ルーチンを終了したのかを情報としてユーザーに伝えるために“ルーチン”がその処理過程で値をセットするものである。
では、その種類と使い方について個々にみていこう。

2.1.1 ユーザーが値をセットするカプセル・ヴァリアブル

ユーザーが前もってその値を設定してやらなければならないカプセル・ヴァリアブルには、次の3種類がある。

L.Abort　1にセットすると、OSI カプセル・ルーチンは、車両に何らかのダメージを受けた場合、処理を中断してメイン・ルーチンに還る。0にセットすると、ダメージを受けてもこれを無視して、処理を続行する。

L.Water　1にセットすると、OSI カプセル・ルーチンは、移動ルーチン処理中、タンクが水域に突き当っても無視して水域内を前進する。0にセットすると、水域を回避するか、もしくは突き当った時点で処理を中断する。

L.Clear　1にセットすると、OSI カプセル・ルーチンは、進路上の障害物を常に迂回して移動する。0にセットすると、障害物を破壊しながら移動し、破壊できない場合は迂回する。

“L.Water” を例に説明を補足すると、“L.Water” が組み込まれている全ての OSI カプセル・ルーチンは、移動中水域に突き当ると “If” コマンドで “L.Water” の値をチェックする。そしてこれが “1” にセットされていれば、水域を障害物とみなさず直進するルーチンに進み、“0” にセットされていればこれを迂回するルーチンに進むよう設計されているのである。

“L.Water” の具体的な効用は、この値を0か1にセットするだけで、OSI カプセル・ルーチンを水陸両用のシャシーとそうでないシャシーの両方に対応させることができる点にある。

2.1.2 ルーチンが値をセットするカプセル・ヴァリアブル

カプセル・ルーチンがその処理過程で値をセットするカプセル・ヴァリアブルには、以下の4種類がある。
これらのヴァリアブルは、それぞれある特定の条件下で値がセットされるように設計されており（例えば“L.EnemyFound?”は、探査ルーチンが敵を発見すると“1”にセットされる）、全ての OSI カプセル・ルーチンは、そこに組み込まれているヴァリアブルの値を、その時の状況に応じて“1”か“0”にセットしてから“Resume”する設計になっている。従って、ユーザーはルーチンが“Resume”した時点でこの値をチェックすれば、同ルーチンがいかなる理由で終了したのかを知ることができる。

L.Damage?	この値は、車両が何らかのダメージを受けたか否かを表わすもので、ダメージを受けると“1”にセットされる。但し、“L.Abort”が“1”に設定されていない場合は、ダメージをチェックすることができない。
L.EnemyFound?	この値は敵を発見したか否かを表わすもので、敵を探知すると“1”にセットされる。“0”の場合はまだ敵を探知していない状態を表わす。
L.EnemyLost?	この値は、敵を見失ってしまったか否かを表わすもので、追跡、または攻撃中の敵がスキャナー・サイトから消えてしまうと“1”にセットされる。“0”の場合は敵の位置を捕捉中であることを表わす。
L.InRange?	この値は、敵が射程内にいるか否かを表わすもので、追跡中の敵を射程内に入れると“1”にセットされ、攻撃中の敵が射程外に出ると“0”にセットされる。

2．1．3 カプセル・ヴァリアブルの使用例

以下はカプセル・ルーチンを組み込んだＡＩプログラムの１例である。

```
        L.Abort = 0
        L.Water = 1
        L.Clear = 1
    Start
        Do Center
        Do Track
        If L.EnemyLost? = 1 then branch to Start
        If L.InRange? = 1 then Do Attack
        Branch to Start

        Include Centerse
        Include Trackene
        Include Normalat
```

※ “Centerse” は、探査ルーチン “Center” がセーブされているファイル名
“Trackene” は、追跡ルーチン “Track” がセーブされているファイル名
“Normalat” は、攻撃ルーチン “Attack” がセーブされているファイル名

> ※ OSI カプセル・ルーチンの仕様に関する解説は、次項を参照のこと。

2.2 ＯＳＩカプセル・ルーチン解説

以下は、OSI 作成のカプセル・ルーチンに関する解説である。これらの中には初歩的なものから非常に高度なものまで、さまざまなものが含まれるが、そのリストを解析して諸君のテクニック向上に役立てるのも１つの利用法である。

カプセル・ルーチン解説の表記について

- １行目はカプセル・ルーチンのファイル名を表わし、( )内はファイル名の意味を表わす。
- ２行目はそのルーチンを呼び出すラベル名である。
- ３行目はそのルーチンで使用されるカプセル・ヴァリアブルの種類を表わす。
- ４行目以降が解説文である。

２.２.１ 探査専用ルーチン

探査用ルーチンは敵を探査しながら戦場を移動するもので、敵を発見した時点でルーチンを終了する。

★ CENTERSE (Center Search)

Center

L.Abort, L.Water, L.Clear, L.EnemyFound?, L.Damage?

このルーチンは、敵を探査しながら戦場の中央(x , y = 31,31)に向かって移動し、移動完了後はそこに静止して敵の探査を続ける。

★ CORNERSE (Corner Search)

Corner

L.Abort, L.Water, L.Clear, L.EnemyFound?, L.Damage?

このルーチンは、現在タンクがいる位置から最も近いコーナーを算出し、敵を探査しながらそのコーナーに向かって移動する。移動完了後は、そこに静止して敵の探査を続ける。

★ EDGESEAR (Edge Search)

Edge

L.Abort, L.Water, L.Clear, L.EnemyFound?, L.Damage?

このルーチンは、戦場の枠づたいを移動しながら敵を探査する。

★ RANDOMSE (Random Search)

Rndsearch

L.Abort, L.EnemyFound?, L.Damage?, L.InRange?

このルーチンは、敵を探査しながら戦場をランダムに歩きまわる。

★ SITSEARC (Sit Search)

SitSearch

L.Abort, L.EnemyFound?, L.Damage?

このルーチンは、一か所に静止したまま敵の探査を続ける。

2.2.2 追跡専用ルーチン

追跡ルーチンは全て探査ルーチンが敵を発見した時点でアクセスされることを前提に設計されており、敵を射程内に入れた時点で終了する。

★ BEELINEH (Beeline Hunt)

Beeline

L.Abort, L.Water, L.EnemyLost?, L.Damage?, L.InRange?

このルーチンはその名が示すように(Beeline:一直線)、敵のいる場所に直行するもので、急接近するときに使用する。但し、破壊不能な障害物に突き当るとそこでルーチンを終了する。

★ TRACKENE (Track Enemy)

Track

L.Abort, L.EnemyLost?, L.Damage?, L.InRange?

このルーチンは、探知した敵が射程内に入ってくるまで一か所に静止して待ち伏せ、一定の時間を過ぎても敵が来ない場合はこちらから接近する。

★ WAITFORE (Wait for Enemy)

Wait

L.Abort, L.EnemyLost?, L.Damage?, L.InRange?

このルーチンは、探知した敵が射程内に入って来るまで、ひたすら一か所に静止してこれを待ち伏せる。

２．２．３ 攻撃専用ルーチン

攻撃ルーチンは敵が射程内に入った時点でアクセスされることを前提に設計されている。

★ BERSERKA (Berserk Attack)
Berserk
なし
敵を１０回連続で砲撃し、ルーチンを終了する。

★ KILLTANK (Killtank)
KillTank
L.Abort, L.Water, L.EnemyLost?, L.Damage?
このルーチンは追跡兼攻撃用ルーチンで、敵がスキャナー･サイトから消えるまで攻撃を続け､敵が射程外に逃げるとこれを追跡する。

★ NORMALAT (Normal Attack)
Attack
L.Abort, L.EnemyLost?, L.Damage?, L.InRange?
このルーチンは敵がスキャナー･サイトから消えるまで攻撃する。

２．２．４ 退却専用ルーチン

★ PANIC (Panic)
Panic
なし
このルーチンは１０hm 後方に撤退する。

PART 5

この章は、チーム・コンバットの設計方法について記載する。

SECTION 1
チーム・コンバット

SECTION BRIEF

OMEGA SYSTEM が行なう戦闘シミュレーションには、オプションとしてチーム・コンバットの機能が備わっている。
この節ではこのチーム・コンバット用のシミュレーションの設計方法、及びチーム・コンバットを行なうサイバータンクが、チーム内で情報の受け渡しをするときに使用する"通信リンク"(Comm-Link) の使用法について解説する。

1.1　チーム・コンバットとは

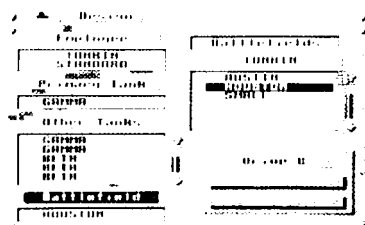
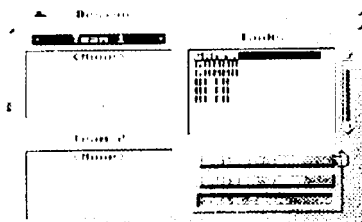
チーム・コンバットとは、複数のサイバータンクが2つのチームに分かれて対戦するもので、1チームは最高7台のタンクで編成される。
勝利の条件は、敵チームの車両を全て破壊することであるが、チーム・コンバットの場合、戦闘方法のもう1つのヴァリエーションとしてシミュレーション・デザイン・ファイル作成の時点で、両チームがそれぞれ戦場のどこかに指令部(HQ：Headquarters)を配置して戦うことができる。この場合勝利の条件は、敵の車両を全て破壊するか、もしくは指令部を破壊するかのいずれかである。

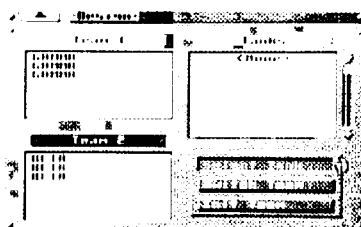
1.2　チーム・コンバット・シミュレーションの設計

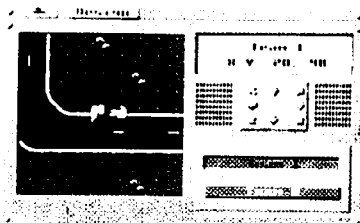
では、チーム・コンバットのシミュレーションの設計について具体例を示しながら解説しよう。

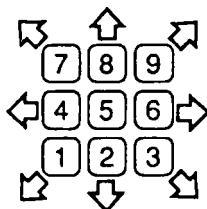
以下は“ＧＡＭＭＡ”３台のチームと、“ＢＥＴＡ”３台のチームを対戦させるシミュレーション・デザイン・ファイルの作成手順である。

1) 通常のシミュレーション・デザイン・ファイル作成と同様、メニュー Simulate の Design a Simulation を選択し、Simulation Design Module に入る。

2) ここで、下図の通り２つのチームを編成するのに必要な全ての車両と戦場を選択する。

3) メニュー Design の Select Teams を選択して下の画面に入り、ここで、タンクのチーム分けを行なう。

※ 2)の画面に戻るときは、Design の Select Cybertank を選択する。

(キーボードの場合) ←,→キーで☜を [Tanks] のウィンド内に合わせ、↑,↓キーで Team 1 に選ぶファイルを反転させ、⏎キーを押す。こうして Team 1 の選択が完了したら次に☜を [Switch Team] に合わせて⏎キーを押し、[Team 2] が反転したら☜をまた [Tanks] のウィンド内に戻して、同様に Team 2 のタンクを選択する。
選択をキャンセルする場合は、☜を削除するファイルに合わせて⏎キーを押す。
(マウスの場合) [Tanks] ウィンド内のファイルを直接クリックして Team 1 のタンクを選択し、次に [Team 2] をクリックしてこれを反転させ、同様にして Team 2 のタンクを選択する。
選択をキャンセルする場合は、削除するファイルを直接クリックする。

尚、[Init Team] (Initialize Team) は、チームに選択した全てのファイルをキャンセルするときに選択する。

4) 指令部を配置する場合は、[Design] の [Position Headquarters] を選択して下の画面に入る。

（キーボードの場合）[←],[→],[↑],[↓]キー、またはテン・キー（下図参照）でまず Team 1 の指令部を配置する。次に [ESC] キーと[↑],[↓]キーで☞を [Team 2] に合わせ、[↵]キーでこれを赤く反転させたら Team 2 の指令部を配置する。

（マウスの場合）画面右の矢印をクリックするか、またはキーボードの操作にならってまず Team 1 の指令部を配置する。次に [Team 2] をクリックしてこれを赤く反転させ、Team 2 の指令部を配置する。

5) メニュー [Design] の [Save Simulation Design] を選んでファイル名を入力したら [Save] を選択する。

以上が、チーム・コンバットのシミュレーション・デザイン・ファイル作成の手順である。

※ 指令部を置かないで戦う場合は 4) を抜かす。

作成したチーム・コンバットのシミュレーション・デザイン・ファイルは通常のシミュレーション・デザイン・ファイルと同じ場所に保管され、その検索、修正、実行も、通常のそれと同じ手順を踏む。
チーム・コンバットの設定が施されているシミュレーション・デザイン・ファイルは、Combat Simulation Module や Cybertank Test Module にかけられると、通常のシミュレーションではなく、自動的にチーム・コンバットを行なうよう設計されている。

ファイル作成の手順を理解したところで、作成時における注意事項について述べておこう。

- 1チームを編成するタンクの数は最高7台とする。
- 2)で Primary Tank 及び、Other Tanks に登録されたタンクは全て Team 1 か Team 2 のいずれかに編入されなければならない。3)の段階で、Tanks の欄に車両を残したままファイルをセーブしてシミュレーションを実行すると、Tanks の欄の車両はどちらのチームにも属さない第3者として独自の行動をとり、チーム間の優劣を評価するうえでの妨げとなる。
- シミュレーション処理中のタンクの“認識番号”は、2)で選択された順序にならう。
- Test Cybertank Module の Trace Mode でプログラムの流れを見たり、Status Mode でユーザー・ヴァリアブルを照会することができるのは、Primary Tank に限られる。

1.3 通信リンクの使用法

Special Items の1つである"Comm-Link"(Communication Link)は、サイバータンクがチーム・コンバットにおいて、同じチーム員と情報交換をするときに使用する通信手段である。
これを装着しなくてもチーム・コンバットは成立するが、これを活用することによってチーム員は遥かに高度な戦術展開を繰り広げることが可能になる。

1.3.1 "Comm-Link"の使用に使われるコマンドの種類

"Comm-Link"の使用に使われるコマンドの種類は以下の5つである。

Switch CommLink on	(通信リンクをオンにする)
Switch CommLink off	(通信リンクをオフにする)
Transmit Code # to team	(暗号"#"をチームに送信せよ)
Copy CommLink Data	(受信したデータをコピーせよ)
Clear CommLink Data	(受信したデータを消去せよ)

解説

Switch CommLink on

"Comm-Link"は、装着しただけで自動的に作動するものではない。チーム員と交信するときは、まずこのコマンドで"Comm-Link"を受信、送信が可能な状態にしなければならない。
送信は、次項の"Transmit Code # to team"コマンドによって行なわれるが、受信は"Comm-Link"がオンになっていれば常時、これを受けることができる。

Transmit Code # to team

● これは送信を処理する唯一のコマンド構造で、“#”の部分にはあらかじめチーム内でその意味を申し合わせておいた“暗号番号”が入る。
● “暗号番号”の決定、及びその定義は全てＡＩ作成者によって任意に行なわれる(その具体的な使用法については後述)。但し“暗号番号”として使用できる数字は 1～100 の間の数字である。
● 送信は常に所属するチームの全員に対して行なわれ、相手を限定することはできない。
● このコマンドは、これによって送信されるデータ項目が全て一律で定められており、或る特定のデータだけを選別して送信するものではない。送信されるデータ項目は以下の通りである。

1) “暗号番号”
2) 暗号を送信したタンクの“チーム員番号”
3) 暗号を送信したタンクの座標位置
4) 暗号を送信したタンクが最後に探知した敵車両の座標位置
5) 敵指令部の座標位置

※ “チーム員番号”とは、シミュレーション・デザイン・ファイル作成時にその車両が所属チームに編入されたときのチーム内での順番にならって決定するものであり（p.244 の 3)の工程）、“認識番号”とは異なる。
※ 3),4) のデータは全て暗号を送信した時点の値である。
※ このコマンドが処理された時点で、敵車両或いは敵指令部が探知できていない場合、それらのｘ座標、ｙ座標の値は共に“０”が送信される。

● 上記のデータは全て、システム・ヴァリアブルによって管理されているものであり、暗号を受信したチーム員の下記のシステム・ヴァリアブルに相当する。

AllyNum	暗号を送信した車両の“チーム員番号”。
AllyCode	送信した“暗号番号”。
AllyX	暗号を送信した車両のその時点のx座標。
AllyY	暗号を送信した車両のその時点のy座標。
AllyEnemyX	暗号を送信した車両が最後に探知した敵のx座標。
AllyEnemyY	暗号を送信した車両が最後に探知した敵のy座標。
EnemyHQX	敵指令部のx座標。
EnemyHQY	敵指令部のy座標。

即ち、“Transmit Code # to team”というコマンドの処理内容は暗号を受信したチーム員の上記のシステム・ヴァリアブルの値をセットすることに他ならない。
そして、これと同時に OMEGA SYSTEM はこれらの値を基にして、更に暗号を受信したチーム員の以下のシステム・ヴァリアブルの値を算出し、これをセットする。

AllyDist	自車両と暗号を送信したチーム員との距離。
AllyEnemyDist	自車両と暗号を送信したチーム員が最後に探知した敵との距離。
AllyEnemyDir	自車両から見た場合の、暗号を送信したチーム員が最後に探知した敵のいる方角。
EnemyHQDist	自車両と敵指令部との距離。

● 以上の値は Cybertank Test Module / Status Mode で一覧できる。

Cybertank Test Module / Status Mode

p.2

Enemy HQ

X Location	--------	EnemyHQX
Y Location	--------	EnemyHQY
Distance	--------	EnemyHQDist

p.3

COMMLINK DATA

Ally Number	--------	AllyNum
Code Received	--------	AllyCode

ALLY TANK (味方車両)

X Location	--------	AllyX
Y Location	--------	AllyY
Distance	--------	AllyDist

ENEMY TANK

X Location	--------	AllyEnemyX
Y Location	--------	AllyEnemyY
Distance	--------	AllyEnemyDist
Direction	--------	AllyEnemyDir

※ 上記 p.3 の9種類のデータを "COMMLINK DATA" と呼ぶ。
※ "COMMLINK DATA" は暗号を受信したときのみセットされるものであり、送信した側が同時に自らの "COMMLINK DATA" をセットすることはない。

Switch CommLink off
このコマンドは“Comm-Link”をオフにするものである。
“Comm-Link”はこれがオンになっている間、チーム員が送信する全ての暗号を無選択に受信するシステムになっており、“COMMLINK DATA”は新しい暗号を受信すると全てのデータが更新されてしまう。
このコマンドは必要な暗号をキャッチしたとき、“Comm-Link”を不通にして、必要なデータが次に送信されてくる別のデータによって書き換えられてしまうのを防ぐためのものである。もちろん“Comm-Link”をオフにしても、最後に記録されたデータは残る。

Copy CommLink Data
このコマンドは、“COMMLINK DATA”の値を Cybertank Test Module / Status Mode / p.4 の“COMMLINK COPY”にコピーするものである。こうしておいてから再び“Comm-Link”をオンにすれば、そのタンクはとりあえず必要なデータを確保した上で、他に重要な暗号が送信されて来ていないかを継続してチェックすることができる。

Clear CommLink Data
これは“COMMLINK DATA”の値を消去するコマンドである。

次は、これらのコマンドが実際のプログラムの中でどのように使われるかを見るが、その前にチーム・コンバットのＡＩを作成する際に踏むべきステップについて述べておこう。

１．３．２ チーム・コンバットにおける戦略の決定

チーム・コンバットのＡＩは、次の事項の決定に沿って立案される。

1) 指令部を配置するか否かの決定
2) チームの戦略の決定
3) チームの構成人員の決定
4) 各チーム員の任務の決定

※ 指令部は１回の攻撃によって破壊されてしまうので、指令部を配置する場合は、この防衛が戦略の決定を大きく左右する。
尚、味方指令部の座標位置は、シミュレーション開始と同時に全チーム員のシステム・ヴァリアブル“AllyHQX” “AllyHQY”にセットされる。

※ 任務の決定は、チーム員全員が汎用性の高い共通のＡＩを持つことによって状況に応じて決まる場合と、初めから別個のＡＩを持たせることによって固定した任務を分担させる場合とがある。

では“Comm-Link”の実際の使用例を、ファイル名“MISSION１”というＡＩのリストの一部を使って見ていこう。

１．３．３ “MISSION１”の基本設計

“MISSION１”は、以下のことを前提に設計されたものであり、

● 指令部を配置する
● チーム員は全て同じＡＩに基づいて行動する
● 最優先攻撃目標を敵指令部に置く

その概要は次の通りである。
各車両は自ら敵指令部、及び敵車両の探査活動を行なうと共に、並行して、敵指令部を発見したときに送信すると定めた暗号“１”または敵車両を発見したときに送信すると定めた暗号“２”が他のチーム員から送信されて来ていないかのチェックを定期的に行なう。

敵指令部の座標位置をつきとめるまでの間は、敵車両の探査、及びこれとの戦闘を行なうが、一旦敵指令部の情報をキャッチした場合は、いかなる状況下にあっても一切これを無視して敵指令部の破壊に急行する。

1.3.4 “MISSION 1” のリストの解説

```
        Main
              Switch CommLink on

              Scan for enemy HQ
①==>          If enemy HQ was found  then  do  HQ/found
②==>          If AllyCode = 1  then  do  HQ/attack
              Scan for enemy tank
③==>          If enemy tank was found  then  do  ET/found
④==>          If AllyCode = 2  then  do  ET/attack

              Rotate scanner left 1
              If tank is aligned with scanner then do Move
              Branch to Main
```

以上は “MISSION 1” のメイン・ルーチンである。
このメイン・ルーチンは、以下にまとめた通り基本的には４つのサブ・ルーチンから構成されている。

①：敵指令部 を発見した場合　サブ・ルーチン “HQ/found” を実行
②：暗号 “1” を受信した場合　サブ・ルーチン “HQ/attack” を実行
③：敵車両　を発見した場合　サブ・ルーチン “ET/found” を実行
④：暗号 “2” を受信した場合　サブ・ルーチン “ET/attack” を実行

では、それらの設計を個々に見ていこう。

★ HQ/found（敵指令部を発見した場合）

```
      HQ/found
⑤==>          Transmit Code 1 to team
      HQ/attack
              If enemy HQ is within range  then  HQ/fire
              If tank not facing enemy HQ then  do HQ/face
              Do Move
              Branch to  HQ/attack
                     ┆
```

このサブ・ルーチンは⑤で即刻、敵指令部発見の暗号“１”を送信して敵指令部の座標位置をチーム員に伝え、自らは敵指令部の破壊に急行する。

★ HQ/attack（暗号“１”を受信した場合）

このサブ・ルーチンは上記“HQ/found”内の“HQ/attack”である。即ち、敵指令部の破壊に急行する。

★ ET/found（敵車両を発見した場合）

```
        ET/found
⑥==>            Transmit Code 2 to team
        ET/f/attack
⑦==>            If  AllyCode = 1  then  HQ/attack
                Scan for enemy tank
                If enemy tank was not found  then  ET/f/resume
                        ⋮
```

このサブ・ルーチンは⑥で即刻、敵車両発見の暗号“２”を送信して敵車両の座標位置をチーム員に伝え、自らは敵車両の破壊に急行する。但し、⑦のコマンドを組み込むことによって敵の追跡、或いは交戦中も絶えず暗号“１”が送信されて来ていないかを定期的にチェックし、これを受信した場合は直ちに“HQ/attack”に分岐する。

★ ET/attack（暗号“2”を受信した場合）

```
       ET/attack
⑧==>           Switch  CommLink  off
⑨==>           Copy  CommLink  Data
⑩==>           Switch CommLink on
⑪==>           Turn tank to  CopyEnemyX  CopyEnemyY
       ET/a/scan
⑫==>           If  AllyCode = 1  then  HQ/attack
               If scanner not aligned then do ET/a/align
               Scan for enemy tank
               If enemy tank was not found then ET/a/resume
                     ⁞
```

このサブ・ルーチンは、入手した敵車両に関するデータが次に送信されて来る別のデータによって書き換えられないように、まず⑧で Comm-Link をオフにし、次に⑨で“COMMLINK DATA”の値を“COMMLINK COPY”に複写してから、最も重要な暗号“1”を見逃さないよう、⑩ですぐに Comm-Link をオンにする。
こうして、当面の攻撃対象である敵車両のデータを確保してから⑪以降でこの車両の破壊に向かうが、この間も⑫で定期的に暗号“1”を受信していないかのチェックを行なう。

1.3.5 暗号番号の使い方

最後に“暗号番号”の使い方について解説しよう。

“Transmit Code # to team”が処理されると、暗号を送信した車両を除く全チーム員の“AllyCode”が、リアルタイムで # の値にセットされることは既に述べた。が、なぜこの処理が自動的に行なわれるのかといえば、それは OMEGA SYSTEM が“AllyCode”という文字列の定義を認識しているからに他ならない。“AllyCode”がシステム・ヴァリアブルと呼ばれる理由はここにある。

しかし、どのような条件の下で、どんな数の暗号を送るかということは、あくまでもＡＩ作成者が決めることであり、敵指令部を発見したときに送信する暗号は何番でなければならないという決まりはない。

例えば、装甲の損傷が 50%を越えた場合“３”という暗号を送信して味方の助けを仰ぐというＡＩを組み込もうと思ったら、暗号“３”を受信した場合に実行されるサブ・ルーチンを設け、その仕様を“３”を送信してきたチーム員の救援に向かうよう設計すればよいのである。

PART 6

この章は、プレー中に表示されうる全てのエラー・メッセージについて記載する。

SECTION 1
エラー・メッセージ一覧

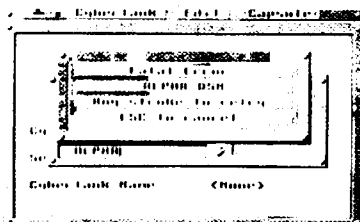
SECTION BRIEF

この節では OMEGA SYSTEM を操作中、問題が生じたときに表示される
さまざまなエラー・メッセージについて解説する。
エラーの種類は大別すると、

- フロッピー・ディスク、及びその操作に起因するもの
- パスワードの管理に起因するもの（セキュリティー・エラー）
- サイバータンクのソース・ファイルの設計内容に起因し、"認定"の過程で発生するもの
- シミュレーション·デザイン·ファイルの設計内容に起因し、戦闘シミュレーションの実行時に発生するもの
- その他

のように分類することができ、発生した個々のエラーに応じてエラーメッセージが表示され、その内容を指摘する仕組みになっている。
では、順番に解説していこう。

1.1　ディスク、及びディスク操作によるエラー

★ Fatal Error： ＊＊＊.＊＊＊

［重大なエラーが］
［ファイル名：＊＊＊.＊＊＊ で発生しました］
［再実行の場合はいずれかのキーを、］
［処理をキャンセルする場合は ESC キーを押してください。］

ディスク、及びディスク操作に起因するエラー・メッセージは全て上のものが表示され、その原因には以下３つのものが考えられる。

- ファイルのセーブを行なおうとしたディスクにライト・プロテクト(書き込み禁止)がかかっている場合

- アクセスしたドライブからディスクが抜かれている場合

- 空き容量"０"（ディスク・フル）のディスクに、更に別のファイルをセーブしようとした場合

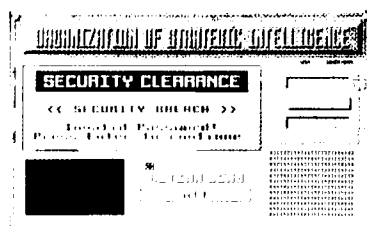
※ 検索したファイルが見つからなかったときも、同じエラー・メッセージが表示されるが、この解説に関してはシミュレーション・エラーの項（P.268）を参照のこと。

エラー・メッセージの２行目には、セーブ、削除、或いはロードに失敗したファイル名が場合に応じて表示される。その拡張子はそれぞれ以下のファイル・カテゴリーを表わす。

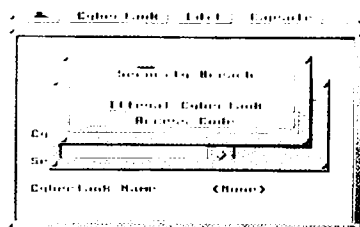
*.DSN —— サイバータンクのソース・ファイル
*.TNK —— サイバータンクの実行ファイル
*.SIM —— シミュレーション・デザイン・ファイル
*.FLD —— 戦場ファイル
*.BLK —— ブロック・ファイル
*.SIP —— シミュレーションの途中データのファイル
*.EIP —— 等級評価の途中データのファイル
*.CAP —— カプセル・ルーチンのファイル

1.2　セキュリティー・エラー

セキュリティー・エラーとは、OMEGA SYSTEM が オメガ・プロジェクトに関する機密を保持するために、パス・ワードのチェックを行ない、これに違反した場合に発生するものである。

★ << SECURITY BREACH >> Invalid Password !

[セキュリティー違反]
[誤ったパスワードが入力されました]
[Enter を押してやり直して下さい]

これは、セキュリティー・クリアランスで誤ったパス・ワードを入力した場合に表示される。

★ Security Breach / Illegal Cybertank Access Code

［セキュリティー違反］
［現在のパスワードではアクセスできません］

● これは他人が作成したサイバータンクのソース・ファイルをロードしようとした場合、もしくはこれをファイル・コピーしようとした場合に表示される。

※ ＡＩプログラムのソース・リストは、個々のエンジニアが培ってきたあらゆるノウハウの結晶である。
OMEGA SYSTEM は､エンジニアの財産とも言えるこのデータをまもるために、ソース・ファイルの開設時に、設計者のパスワードをファイルの一部に書き込むことで、同ファイルが誰によって作成されたものかを管理しており、第３者がこれと異なるパスワードでアクセスしてきた場合には、リストのロード、及びコピーを拒否するシステムになっている。

1.3 認定エラー

以下に記載するのは Cybertank Authorization Module で、認定処理の過程において発生しうるエラーである。
(配列はアルファベット順で、冠詞を除く)

★ The cost of your tank's chassis exceeds your available budget. Please reduce the size of your chassis.
[あなたのタンクのシャシーは、あなたに割り当てられた予算の枠を超えています。シャシーの総計額を減らして下さい]
このエラーは、あなたが、あなたより等級の高いエンジニアからコピーしたサイバータンクのソース・ファイルを認定処理にかけた際、シャシーがあなたの予算枠を超過している場合に発生する。

★ '* * *' does not compute in the line:
[次の行の中で使われている '* * *' は処理できません]
このエラーは、スペル・ミスやスペースの入れ忘れなどで、CCL リザーブ・ワードにない単語が使用された場合に発生する。

★ Duplicate Label Found: '* * *'
[ラベル '* * *' が重複して使われています]
このエラーは、プログラム内に同名のラベルが2回以上使用されている場合に発生する。

★ The following capsule was not found '* * *'
[次のカプセルは発見されませんでした]
このエラーは、同じディレクトリ上に存在しないカプセル・ルーチンを“Include”しようとした場合に発生する。
但し、この場合このメッセージにさきがけて“Fatal Error:”の表示が出る。(p.268 参照)

★ The following label is too long: [次のラベルは文字数超過]
このエラーは、１４文字以上のラベルが存在する場合に発生する。
ラベル名は１３文字以内でなければならない。

★ The following line is an illegal label:
[次の行のラベル名には誤りがあります]
このエラーは、ラベルにスペースが入っている場合や、コマンドが誤って左端から始まっている場合に発生する。

★ If you include a capsule, that capsule cannot include another.
[カプセル・ルーチンを Include する場合、そのカプセルが更に他のカプセル・ルーチンを Include することはできません]
他のカプセル・ルーチンを Include した構造をもつカプセル・ルーチンをＡＩに組み込む場合、Include コマンドは使えない。
そのリストを書き込むことで対処する。

★ The label '* * *' is referenced but not defined.
[ラベル '* * *'に言及していますが、同ラベルは未定義です]
このエラーは、ＡＩの中のあるシーケンス・コマンドが存在しないラベルへの分岐、或いは実行を指示している場合に発生する。

★ Unknown Command： ＊＊＊ in：

［次の行の中で使われている ＊＊＊ は、不明コマンドです］

このエラーは、コマンド・ラインから動詞に相当する単語が抜けている場合や、（例）Move tank forward 1 → tank forward 1

ラベル数が１１３個以上に及んだ場合に発生する。

後者の場合は、ラベル数が１１２個以内に納まるようプログラムを組み直す。

★ User Variables Exceeded.［ユーザー・ヴァリアブルの使用超過］

このエラーは、３３個以上のユーザー・ヴァリアブルを使用した場合に発生する。この場合、ユーザー・ヴァリアブルが３２個以内に納まるようプログラムを組み直す。

★ Value used is out of range in following line：

［次の行の中で使われている設定値は値域外です］

このエラーはコマンドの中の設定値が、そのコマンドのとりうる値でない場合に発生する。

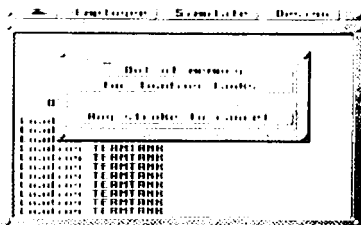
★ You are missing one or more items from your tank's chassis. Please complete your tank before authorizing it.

［タンクのシャシーに、１ないし複数のコンポーネントの装備漏れがあります。認定を受ける前にタンクを完成させて下さい］

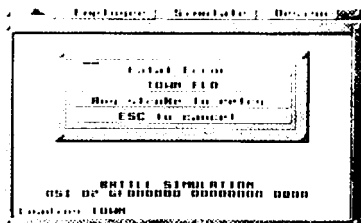
★ Your tank must be given intelligence before it can be authorized.

［あなたのサイバータンクは、認定を受ける前にＡＩプログラムを作成しなければなりません］

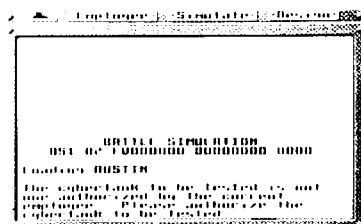
1.4　シミュレーション・エラー

★ Out of memory for loading tanks

［メモリー容量超過］
［サイバータンクをロードすることができません］
［いずれかのキーを押してキャンセルして下さい。］

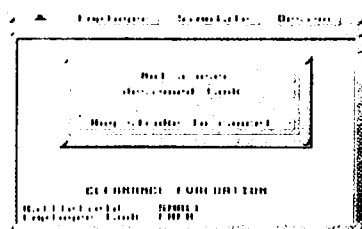
Combat Simulation Module 及び Cybertank Test Module は、シミュレーションを処理できるメモリーの容量に限りがある。
このメッセージが表示された場合は、シミュレーション・デザイン・ファイルに登録されたサイバータンクの数を減らすか、或いはＡＩプログラムを短縮することで対処する。

★ Fatal Error： ***.***

［重大なエラーが］
［ファイル名 ***.*** で発生しました］
［再実行の場合はいずれかのキーを、］
［処理をキャンセルする場合は ESC キーを押して下さい。］

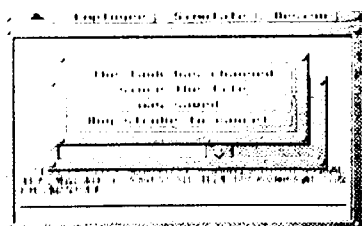
このエラーは、シミュレーションを処理する際に必要とするファイルがディスク上に見つからなかった場合に発生し、メッセージの２行目が見つからなかったファイル名を表わす。
(ファイル名の拡張子については p.263 を参照のこと)
Combat Simulation Module 及び Cybertank Test Module が戦闘シミュレーションを処理する場合、シミュレーション･デザイン･ファイルに登録されている自車両、敵車両、戦場の各ファイルは必ず同一ディスクの同一ディレクトリ上になければならない。
このエラーの原因は、シミュレーション･デザイン･ファイル作成の際、そこで選択した他のディスク上のファイルをあなたのＩＤディスクにコピーし忘れたものと考えられる。
(ファイル・コピーの方法は p.165 を参照)

★ The cybertank to be tested is not one authorized by …

[検査するサイバータンクは、あなたが]
[認定処理にかけたものではありません。]
[検査するサイバータンクを認定処理にかけて下さい。]

このエラーは、他のＩＤディスクからコピーした実行ファイルを Primary tank とするシミュレーション・デザイン・ファイルを Cybertank Test Module にかけた場合に発生する。これを解決するには、同サイバータンクのソース・ファイルを入手し、これをあなたが改めて認定処理しなければならない。

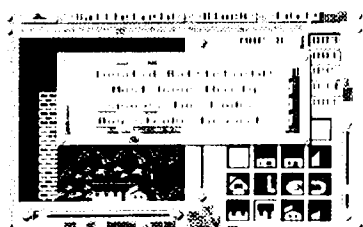
★ Not a user designed tank.

[あなたが設計したタンクではありません。]
[いずれかのキーを押してキャンセルして下さい。]

これは他のディスクからコピーした実行ファイルを、等級評価にかけた場合に発生する。

★ The tank has changed since the file was saved

[当ファイルがセーブされた後に]
[実行ファイルが変更されました]

Clearance Evaluation Module 及び、Combat Simulation Module が処理した戦闘シミュレーションの途中データを記録したファイルを呼び出した際、そこでロードされるタンクの実行ファイルのデータが、前者のファイルがセーブされた時点と異なる場合にこのエラーは発生する。

1.5　その他

★ Invalid Battlefield !

［戦場の設計に誤りがあります］
［タンクを配置する３０のスペースを確保して下さい。］
［いずれかのキーを押してキャンセルして下さい。］

これは、Design Battlefield Module で作成した戦場に、障害物の無いセルが３０以上含まれない場合に発生する。
シミュレーションの開始時点で、Combat Simulation Module は、無作為に全サイバータンクのスタート位置を選定するが、サイバータンクを配置できるのは、障害物のないセル上に限られている。このため、戦場を作成するときは、最低３０の障害物の無いセル（タイプ·ナンバー１，３）が含まれていなければならない。

付録 1
ＣＣＬコマンドに使用される単語一覧

align	aligned	ally	are
armor	at	available	backward
beep	being	beyond	branch
break	clear	closest	code
CommLink	copy	data	destruct
detect	direction	distance	do
down	empty	enemy	face
facing	fire	for	forward
found	fuel	functional	get
gosub	goto	HQ	if
include	internal	is	jam
key	kit	last	launch
left	lock	locked	lower
move	movement	not	object
obstructed	obstruction	off	on
pressed	raise	random	range
remaining	remote	repair	resume
right	rotate	scan	scanned
scanner	self	shield	signal
switch	tank	team	then
to	transmit	treads	turn
unavailable	unlock	unlocked	up
was	weapon	with	within

付録 2
システム・ヴァリアブル一覧

AllyCode	AllyDir	AllyDist
AllyEnemyDir	AllyEnemyDist	AllyEnemyX
AllyEnemyY	AllyHQX	AllyHQY
AllyNum	AllyX	AllyY
ArmorDamage	CopyCode	CopyDir
CopyDist	CopyEnemyDir	CopyEnemyDist
CopyEnemyX	CopyEnemyY	CopyNum
CopyX	CopyY	EnemyDist
EnemyHQDist	EnemyHQX	EnemyHQY
EnemyX	EnemyY	FuelLevel
IntDamage	KitsLeft	ObjDist
ObjType	ObjX	ObjY
ObstacleDist	ObstacleType	ObstacleX
ObstacleY	RandomNum	RemotesLeft
ScanDamage	ScanDir	TankDir
TankNum	TankX	TankY
TreadDamage	WeapDamage	XYDist

以下に記載するのは、システム・ヴァリアブルの全用語の解説である。

AllyCode　：チーム員が送信した暗号。値域は 0～100。

AllyDir　：暗号を送信したチーム員のいる方角。

AllyDist　：暗号を送信したチーム員と自車両との距離。

AllyEnemyDir ：自車両から見た場合の、暗号を送信したチーム員が最後に探知した敵車両のいる方角。

AllyEnemyDist：自車両と暗号を送信したチーム員が最後に探知した敵車両との距離。

AllyEnemyX ：暗号を送信したチーム員が最後に探知した敵車両のx座標。値域は 1～62。

AllyEnemyY ：暗号を送信したチーム員が最後に探知した敵車両のy座標。値域は 1～62。

AllyHQX ：味方司令部のx座標。

AllyHQY ：味方司令部のy座標。

AllyNum ：暗号を送信したタンクのチーム員番号。
（チーム員番号については、p.249 参照）

AllyX ：暗号を送信したチーム員のx座標。

AllyY ：暗号を送信したチーム員のy座標。

ArmorDamage ：装甲の現在の損傷比率。値域は 0～100。
この値が 100 になるとサイバータンクは破壊される。

CopyCode ："Copy CommLink Data" が処理されたときの "AllyCode" の値。

CopyDir ："Copy CommLink Data" が処理されたときの "AllyDir" の値。

CopyDist　　　：“Copy CommLink Data” が処理されたときの
“AllyDist” の値。

CopyEnemyDir ：“Copy CommLink Data” が処理されたときの
“AllyEnemyDir” の値。

CopyEnemyDist：“Copy CommLink Data” が処理されたときの
“AllyEnemyDist” の値。

CopyEnemyX　：“Copy CommLink Data” が処理されたときの
“AllyEnemyX” の値。

CopyEnemyY　：“Copy CommLink Data” が処理されたときの
“AllyEnemyY” の値。

CopyNum　　　：“Copy CommLink Data” が処理されたときの
“AllyNum” の値。

CopyX　　　　：“Copy CommLink Data” が処理されたときの
“AllyX” の値。

CopyY　　　　：“Copy CommLink Data” が処理されたときの
“AllyY” の値。

EnemyDist　　：最後に探知した敵車両までの距離。

EnemyHQDist　：敵司令部までの距離。

EnemyHQX　　：敵司令部のｘ座標。

EnemyHQY　　：敵司令部のｙ座標。

EnemyX　　　：最後に探知した敵車両のｘ座標。

EnemyY　　　：最後に探知した敵車両のｙ座標。

FuelLevel　　：燃料の残量。値域は、0～100。

IntDamage　　：電気系統の現在の損傷比率。値域は 0～100。
　　　　　　　この値が 100 になるとサイバータンクは破壊される。

KitsLeft　　：修理用キットの残り数。

ObjDist　　　：スキャナーが最後に探知した物体までの距離。

ObjType　　　：スキャナーが最後に探知した物体のタイプ・ナンバー。

ObjX　　　　：スキャナーが最後に探知した物体のｘ座標。値域は
　　　　　　　0～64。

ObjY　　　　：スキャナーが最後に探知した物体のｙ座標。値域は
　　　　　　　0～64。

ObstacleDist ：移動センサーが、最後に探知した障害物までの距離。
　　　　　　　値域は 0～3。

ObstacleType ：移動センサーが、最後に探知した障害物のタイプ・ナンバー

ObstacleX　　：移動センサーが、最後に探知した障害物のｘ座標。
　　　　　　　値域は 0～64。

ObstacleY　　：移動センサーが、最後に探知した障害物のｙ座標。
　　　　　　　値域は 0～64。

RandomNum　：“Get Random [to #]”コマンドで得たランダム値。

RemotesLeft　：リモート・スキャナーの残り数。

ScanDamage　：スキャナーの現在の損傷比率。値域は1～100。
この値が 100 になるとスキャナーが破壊され、サイバータンクは目標物を探知できなくなる。

ScanDir　：スキャナーが現在向いている方角。値域は0～7。

TankDir　：自車両の現在の向き。値域は0～7。

TankNum　：自車両の認識番号。(認識番号については p.64 参照)

TankX　：自車両のｘ座標

TankY　：自車両のｙ座標

TreadDamage　：トレッドの現在の損傷比率。値域は 1～100。
この値が 100 になるとトレッドが破壊され、サイバータンクは移動、方向転換ができなくなる。

XYDist　：“Get Distance to X Y”コマンドが算出した座標点(ｘ，ｙ)までの距離。

WeapDamage　：武器の現在の損傷比率。値域は 1～100。
この値が 100 になると武器が破壊され、サイバータンクは攻撃ができなくなる。

付録 3
CCLコマンド一覧早見表

※ 表記の中で使用される記号、略号は p.193 の規約に準じる。

★タンクの移動

コマンド	ページ
Move [tank] forward #	(p.142)
Move [tank] backward #	(p. 〃)
Turn [tank] left #	(p.143)
Turn [tank] right #	(p. 〃)
Turn [tank] to #	(p. 〃)
Turn [tank] to X Y	(p. 〃)
Turn [tank] to face [enemy] tank	(p.144)
Turn [tank] to face enemy HQ	(p. 〃)
Align tank [with scanner]	(p. 〃)
If tank [is](not) aligned[with scanner] then <seq.com>	(p.150)
If [movement is] (not) obstructed then <seq.com>	(p.149)
If obstruction [is] enemy HQ then <seq.com>	(p.197)
If obstruction [is] ally HQ then <seq.com>	(p. 〃)
If tank [is] (not) facing [enemy] tank then <seq.com>	(p.150)
If tank [is] (not) facing enemy HQ then <seq.com>.	(p. 〃)
If tank [is] (not) facing X Y then <seq.com>	(p.198)
Detect [obstruction] at #	(p.144)
Detect [obstruction] at tank direction	(p. 〃)
Detect [obstruction] at scanner direction	(p. 〃)
If [tank] treads [are](not) functional then <seq.com>	(p.150)

★スキャナーの使用

Command	Page
Rotate [scanner] left #	(p. 145)
Rotate [scanner] right #	(p. 〃)
Rotate [scanner] to #	(p. 〃)
Rotate [scanner] to X Y	(p. 146)
Rotate [scanner] to face [enemy] tank	(p. 〃)
Align scanner [with tank]	(p. 〃)
Scan for [enemy] tank	(p. 145)
Scan for [closest] object	(p. 〃)
Scan for enemy HQ	(p. 〃)
Lock [scanner]	(p. 148)
Unlock [scanner]	(p. 〃)
Jam [scanner signal]	(p. 〃)
Launch [remote scanner]	(p. 〃)
If scanner [is](not) functional then <seq.com>	(p. 150)
If scanner [is](not) aligned[with tank] then <seq.com>	(p. 〃)
If [enemy] tank [was] (not) found then <seq.com>	(p. 149)
If [closest] object [was] (not) found then <seq.com>	(p. 〃)
If enemy HQ [was] (not) found then <seq.com>	(p. 〃)
If [closest] object [is] enemy HQ then <seq.com>	(p. 202)
If [closest] object [is] ally HQ then <seq.com>	(p. 〃)
If [scanner is] locked then <seq.com>	(p. 150)
If [scanner is] unlocked then <seq.com>	(p. 〃)
If [tank is] (not) being scanned then <seq.com>	(p. 205)
If remote [scanner is] available then <seq.com>	(p. 151)
If remote [scanner is] unavailable then <seq.com>	(p. 〃)

★武器の使用

Fire [weapon] at [enemy] tank	(p. 147)
Fire [weapon] at [closest] object	(p. 〃)
Fire [weapon] at obstruction	(p. 〃)
Fire [weapon] at X Y	(p. 〃)
Fire [weapon] at tank direction	(p. 〃)
Fire [weapon] at scanner direction	(p. 〃)
Fire [weapon] at enemy HQ	(p. 〃)
If weapon [is] (not) functional then <seq.com>	(p. 150)
If [enemy]tank [is]within [weapon]range then <seq.com>	(p. 149)
If [enemy]tank [is]beyond [weapon]range then <seq.com>	(p. 〃)
If [closest]object[is]within[weapon]range then <seq.com>	(p. 〃)
If [closest]object[is]beyond[weapon]range then <seq.com>	(p. 〃)
If enemy HQ [is] within [weapon] range then <seq.com>	(p. 〃)
If enemy HQ [is] beyond [weapon] range then <seq.com>	(p. 〃)

★ディフェンス・シールド

Raise [shield]	(p. 148)
Lower [shield]	(p. 〃)
If [shield is] up then <seq.com>	(p. 151)
If [shield is] down then <seq.com>	(p. 〃)

★タンクの修理

Repair Internal	(p. 149)
Repair Armor	(p. 〃)
Repair Treads	(p. 〃)
Repair Scanner	(p. 〃)
Repair Weapon	(p. 〃)
If [Repair] Kit [is] available then <seq.com>	(p. 151)
If [Repair] Kit [is] unavailable then <seq.com>	(p. 〃)

★通信リンクの使用

Transmit [Code] # [to Team]	(p. 223)
Clear [CommLink] Data	(p. 〃)
Copy [CommLink] Data	(p. 〃)
Switch [CommLink] on	(p. 〃)
Switch [CommLink] off	(p. 〃)

★手動制御

If [last] key [pressed] then <seq.com>	(p. 220)
If [last] key [pressed] = A～Z then <seq.com>	(p. 220)

★カプセル・ルーチンの使用

Include “***”	(p. 222)

★シーケンス・コマンド

Branch to “Label”	(p. 221)
Goto “Label”	(p. 〃)
Do “Label”	(p. 〃)
Gosub “Label”	(p. 〃)
Resume	(p. 〃)

★その他のコマンド

Self Destruct	(p. 148)
Get Distance [to] X Y	(p. 215)
Get Random	(p. 216)
Get Random to #	(p. 〃)
Beep	(p. 217)
Break	(p. 219)
*	(p. 218)

索引

※ 項目は英文と和文に分けて記載した。頭に英文字の付く語（ex. ＡＩボード）は、英文に分類した。

★ 英文（アルファベット順）

★ 和文（５０音順）

●宛先　〒113 東京都文京区本駒込6-14-9 東書第2ビル
トンキンハウス・ユーザーサポート 係

☎03-3942-4004
PM2:00～PM5:00
(土曜、日曜、祭日は除きます)